# Robot intelligent télécommander par un réseau neuronal en utilisant des images réelles

Joseph Sayegh

**CONSERVATOIRE NATIONAL DES ARTS ET METIERS**

**PARIS**

---

**MEMOIRE**

**présenté en vue d'obtenir**

**le DIPLOME d'INGENIEUR CNAM**

**SPECIALITE : ELECTRONIQUE**

**par**

**Joseph Sayegh**

# Robot Intelligent Télécommander Par Un Réseau Neuronal En Utilisant Des Images Réelles

**Soutenu le 11 novembre 2010**

---

**JURY**

**PRESIDENT : Michel TERRE**

**MEMBRES : Chaouki DIAB, Khaled ITANI, Haissam HAJJAR, Yasser MOHANNA**

# Remerciements

J’adresse mes plus vifs remerciements  au jury, Président et membres, pour avoir bien voulu examiner ce travail, pour leur temps et leurs conseils précieux.

Je tiens aussi à exprimer ma profonde gratitude pour l’aide et les conseils que mon professeur tuteur Dr. Chaouki DIAB a bien voulu m’accorder.

Mes remerciements s’adressent aussi au Conservatoire National des Arts et Métiers (CNAM) pour la chance qu’il m’a donnée pour devenir l’un de ses ingénieurs-diplômés.

Un grand merci à toute personne qui a contribué de près ou de loin à l’élaboration de ce travail.

# RÉSUMÉ

Ce document décrit le travail que nous avons effectué pour étudier et développer un prototype d'un robot autonome capable de se déplacer dans un environnement réel et contrôlable par un réseau neuronal.

Il décrit en détails la conception de la partie électronique et la construction de la partie mécanique du robot ainsi que le développement, sous Matlab, du programme qui inclut toutes les fonctions et les interfaces graphiques nécessaires pour tester et évaluer la capacité d'un réseau neuronal à naviguer un robot en utilisant des images réelles.

A la fin de ce document, nous présentons les résultats obtenus et l'évaluation de l'essai du prototype dans un environnement réel mais à contraintes limitées.

# ABSTRACT

This paper describes the work we have done to study and develop a prototype of an autonomous robot capable of moving in a real environment and controlled by a neural network.

It covers in detail the design of the electronics, the construction of the mechanical part of the robot and the program development in Matlab which includes all the necessary functions and the graphical user interfaces to test and evaluate the capability of a neural network to navigate a robot using real images.

At the end of this paper, we present the obtained results and the evaluation of the system test in a real but limited environment.

# SOMMAIRE

# PARTIE II

## Chapitre 4 PARTIE MECANIQUE



## Chapitre 5 PARTIE ELECTRONIQUE



## Chapitre 6 PARTIE LOGICIELLE

# PARTIE III

## Chapitre 7 TEST ET EVALUATION



---

# INTRODUCTION

Il est indéniable que les outils de simulation sont devenus incontournables dans différents domaines, particulièrement dans le champ de Recherche et de Développement « R&D », soit pour leur capacité de prévoir les résultats d'une expérience ou d'un modèle conçu en créant son propre environnement virtuel, soit pour réduire le risque en cas d'un environnement dangereux. Dans les deux cas le facteur économique est fortement considéré.

Pourtant, il est parfois nécessaire de réaliser une simulation dans un contexte réel, par exemple, en cas d'évaluation des résultats dans un évènement à temps réel, ou durant le test d'un prototype.

L'objectif de ce projet est de réaliser un robot expérimental qui constitue une liaison entre l'environnent virtuel et celui du temps réel. Le prototype développé à travers ce projet peut servir comme outil de développement et de simulation faisant appel à des applications robotiques. Les éventuelles applications visées incluent toute application invoquant la mobilité dans sa caractéristique. Nous citons les applications industrielles, militaires et domestiques dans des domaines tels que le transport, la télé-guidance ou l'auto-guidance, le sondage à distance, … etc.

Pour tester notre robot-prototype, nous avons choisi d'examiner la capacité d'un réseau neuronal à faire naviguer une machine d'une manière autonome en se servant du traitement des images en temps réel.

Le mémoire est divisé en trois parties :

1. La PARTIE I est divisée en trois chapitres: Le 1$^{er}$ chapitre est consacré au cahier des charges dans lequel nous décrivons les spécifications et les contraintes sur lesquelles nous nous sommes basés pour la conception du robot. Ensuite, les 2 autres chapitres constituent une synthèse bibliographique sur les thèmes de la robotique et des réseaux de neurones.

2. La PARTIE II se décline en trois chapitres aussi, à travers lesquels nous présentons le travail réalisé dans tous ses aspects mécanique, électronique et informatique.

3. La PARTIE III s'occupe des tests et de l'évaluation des performances effectués ainsi que des résultats obtenus, avant de conclure et d'établir les perspectives.

# PARTIE I

## Chapitre 1 CAHIER DES CHARGES

## Chapitre 2 NOTIONS EN ROBOTIQUE

## Chapitre 3 NOTIONS SUR LES RESEAUX DE NEURONES

# Chapitre 1 CAHIER DES CHARGES

## 1.1. Introduction:

La définition du cahier des charges est la première étape dans l'élaboration et la mise en œuvre d'un projet aussi grand que la conception et la réalisation d'un robot de performances aussi modestes qu'elles soient.

Dans ce chapitre nous allons choisir les spécifications du robot en répondant aux questions suivantes :

- Quel type de robot a-t-on besoin? Choisissons-nous un type autonome ou télé-opéré ? À quel type de moteur et de quelle puissance doit-on avoir recours ?
- Quel type de batterie doit-on utiliser ?
- Quels types de détecteurs et de capteurs doivent équiper ce robot?

Les réponses sur les dernières quatre questions dépendent d'une manière ou d'une autre de la réponse à la première.

## 1.2. Le type du robot :

Les robots ne sont pas censés nécessairement se déplacer autour d'un plancher. Certains sont conçus pour rester sur place et manipuler des objets placés autour d'eux.

En fait, en dehors des laboratoires de recherche et des amateurs de garage, les types les plus communs des robots, sont ceux utilisés dans la fabrication. En général, ces robots sont fixes. Ils aident à fabriquer des automobiles, appareils, et même d'autres robots!

Par contre, les robots mobiles sont conçus pour se déplacer d'un endroit à un autre, en utilisant des roues, des rails ou des jambes. Ils peuvent également être équipés d'un ou de plusieurs bras leur permettant de manipuler des objets se trouvant sur leurs trajets.

En principe, les robots stationnaires exigent généralement une plus grande précision, de la puissance et de l'équilibre, puisqu'ils sont conçus pour saisir et lever des objets généralement lourds. De même, les robots mobiles ont leurs propres contraintes liées à leur degré de manœuvre, à l'alimentation adéquate dont ils ont besoin, … etc.

Pour tester la possibilité de naviguer, en utilisant une caméra et un logiciel de simulation d'un réseau de neurones, nous avons choisi de construire un robot mobile sur roues, dont l'architecture lui permettrait de s'équiper par d'autres détecteurs et/ou capteurs en fonction des situations et des applications dans lesquelles on souhaitera l'expérimenter.

Or, pour pouvoir utiliser un robot autonome dans diverses situations et pour plusieurs types

d'applications, il faut lui réaliser, à chaque fois, des circuits sur mesure adaptés à l'application visée, ce qui n'est pas du tout commode.

En fait, cette contrainte serait imposée par le souci de lui faire garder une parfaite autonomie.

Ceci devient d'autant plus onéreux que si un effort de développement logiciel serait encore nécessaire pour l'adapter à chaque nouvelle application, comme c'est le cas dans notre projet.

De l'autre côté, les robots télé-opérationnels sont plus flexibles; en effet, par un simple moyen de communication bidirectionnelle entre le robot mobile et un ordinateur, ce dernier peut assurer le pilotage de l'engin, grâce d'une part, à un logiciel de simulation dont la structure générale peut être commune à tout type d'application, et d'autre part aux informations captées et mesurées par les instruments de bord.

Les décisions prises par le logiciel seront ensuite traduites par des commandes émises au robot et exécutées par les organes de la partie opérative.

Ce type d'architecture relativement souple offre à notre système la possibilité de s'adapter facilement à des applications diverses en ajoutant presque exclusivement les capteurs et détecteurs convenables.

Enfin, notons que, puisque la présence d'un câble de communication gêne fortement le mouvement du robot, le choix d'une communication hertzienne sans fil s'impose.

### *1.3. Les types de moteur :*

Les moteurs DC sont les piliers de la robotique. De petites dimensions, ce genre de moteur, lorsqu'il est connecté aux roues et grâce à un système de réduction « engins », peut mettre en marche un robot de 10, 20 voire 40 kg. Par une chiquenaude d'un commutateur, un simple clic de relais, ou une tique d'un transistor, le moteur s'arrête et bouge dans une autre direction. Un simple circuit électronique nous permet un contrôle facile de la vitesse, d'une lente rampe à un mouvement rapide.

Les moteurs à courant continu ne sont pas chers, ils peuvent livrer une bonne force de couple par rapport à leurs petites tailles, et sont facilement adaptables à une variété de robot.

Par leur nature, les moteurs DC sont plutôt imprécis. Sans un servomécanisme de rétroaction ou compte-tours, il est pénible de déterminer à quelle vitesse un moteur DC tourne. En outre, il est difficile de commander le moteur pour lui faire tourner un nombre précis de révolutions. Pourtant, c'est exactement l'exigence des certains genres de robots, en particulier ceux qui sont équipés par un bras.

Les moteurs pas à pas (figure 1.1) sont, en effet, des moteurs DC avec un plus. Au lieu d'être mis en marche par un flux continu de courant comme c'est le cas d'un moteur DC régulier, ils sont contrôlés par des impulsions électriques. Chaque impulsion fait tourner l'arbre du moteur d'un angle déterminé. Plus des impulsions sont envoyées au moteur, plus l'arbre tourne.

Bien qu'ils sont plus coûteux, les moteurs pas à pas ne sont pas faciles à utiliser comme les moteurs DC standard; toutefois, ils apportent des solutions à beaucoup de problèmes avec les moindres

complications possibles.


**Figure 1.1,** Le moteur pas à pas « SY35ST28-0504A » de 200 pas par tour.

Or, pour avoir une action bien contrôlée, il ne suffit pas d'en faire la commande; il faut aussi vérifier qu'elle a été correctement exécutée. Sinon, l'ensemble des opérations qui suivent peuvent être erronées.

En effet, si pour une raison ou une autre, le moteur pas à pas n'a pas pu effectuer le mouvement commandé, il faut que le système de pilotage en soit averti. C'est le rôle du servomécanisme (figure 1.2) qui offre un système de rétroaction. La sortie du moteur est couplée à un circuit de contrôle, La vitesse de déplacement et/ou de la position sont reliées et contrôlées par ce circuit. Si pour une raison quelconque la rotation du moteur est interrompue, le mécanisme de rétroaction détecte que la sortie du moteur n'est pas encore dans l'emplacement souhaité. Alors, il continue à corriger l'erreur jusqu'à que le moteur atteint son bon point.

Pour notre système, nous cherchons à avoir un moteur léger et à faible consommation d'énergie fournissant une bonne force de couple de 2 N.m (le choix de ce couple sera justifié par un calcul décrit dans le chapitre 4 « Partie Mécanique »). La précision n'est pas un facteur obligatoire lorsqu'il s'agit de faire naviguer un robot à quatre roues. Par conséquent, notre choix se portera sur un moteur DC. Si une application future exige la connaissance de la vitesse ou la position du moteur, un compte-tours peut être ajouté sur l'arbre du moteur.

**Figure 1.2,** Le servomoteur « HS-65HB »

### *1.4. Le type de batterie:*

Plusieurs types de batteries sont disponibles sur le marché. Nous sommes intéressés par les types rechargeables.

NICKEL-CADMIUM ou Ni-Cad, sont parmi les moins coûteux et les plus faciles à obtenir. Mais elles sont de faible autonomie et souffrent d'effet de mémoire qui réduit leur capacité utile de stockage, et qui se produit lorsque la cellule n'est pas complètement déchargée avant qu'elle soit rechargée de nouveau.

NICKEL METAL HYDRIDE ou NiMH, sont de taille et poids similaires aux Ni-Cad, mais ayant une capacité d'environ 50% supérieure à celle du Ni-Cad.

Étant donné qu'elles ont une faible résistance interne, les NiMH fonctionnent mieux lorsqu'elles sont utilisées à courant élevé. Contrairement aux Ni-Cad, elles ne souffrent pas de l'effet de mémoire. Les batteries NiMH doivent être rechargées par des chargeurs spécifiques qui peuvent effectuer la recharge à un taux agressif pendant une heure ou deux.

Ces batteries sont capables, malgré leur légèreté, de fournir un courant important, mais à un prix relativement excessif dès qu'on utilise plus d'une cellule.

Les batteries du type lithium-ion sont très utilisées dans les ordinateurs portatifs. Elles sont les mieux utilisées à une décharge constante ce qui les rend relativement plus coûteuses. Elles fournissent la plus haute densité d'énergie. Comme d'autres piles rechargeables, elles exigent leurs propres circuits de recharge.

Batterie à plomb (figure 1.3), sont des batteries puissantes et lourdes, souvent utilisées comme sauvegarde ou alimentation de secours pour les ordinateurs, les lumières, et le matériel téléphonique. Les batteries de motocycle sont considérées comme une bonne cellule de puissance pour les robots. Faciles à obtenir, compactes et relativement légères, elles fournissent un haut courant durant un temps raisonnable, ce qui les rend parfaites pour des applications robotiques.


**Figure 1.3,** Batterie à plomb 12V d'une capacité égale à 90Ah

En conséquence, les batteries batterie à plomb restent une bonne solution du fait qu'elles délivrent une bonne densité d'énergie à prix très abordable. Leur principal inconvénient reste leur poids relativement lourd. Malgré cela, nous avons choisi ce type de batterie pour assurer l'alimentation de notre prototype. C'est un compromis que nous avons bien voulu accepter.

### *1.5. Le choix de détecteurs:*

En bref, nous parlerons ici principalement de la détection des obstacles par des capteurs de type "non contact". Deux types sont  considérés: La détection par infrarouge et la détection par ultrasons.

La lumière propage toujours en ligne droite, mais elle rebondit presque sur tout objet ou obstacle. On peut utiliser cette méthode qui est à notre avantage. Il s'agit de construire une détection de collision par infrarouge. On peut monter plusieurs capteurs à infrarouge à la périphérie du robot. Ils peuvent fournir des détails sur l'environnement extérieur à un ordinateur ou au circuit de contrôle.

Notons que les objets réfléchissent la lumière de différentes façons. Nous devrons probablement ajuster la sensibilité des détecteurs afin d'améliorer leur fiabilité que ce soit pour un obstacle à facette lisse et clair tel qu'un mur blanc ou un obstacle à facette rugueuse et foncée tel qu'un fauteuil marron foncé.

De plus, le détecteur d'infrarouge devrait être isolé de la lumière ambiante ainsi que de la lumière directe provenant de l'émetteur à base de "LED". Le positionnement de la LED est primordial, il faut en faire attention durant l'alignement du pair émetteur/récepteur.

Comme la lumière, le son se propage dans des lignes droites et rebondit sur tout objet se trouvant sur son trajet. Les ultrasons (de fréquences supérieures aux fréquences audibles) peuvent être utilisés à la base des détecteurs de proximité des objets ainsi que pour mesurer la distance séparant ces objets du détecteur.

L'onde ultrasonore est émise par un transducteur, réfléchie par un objet, puis reçue par un autre transducteur. L'avantage des détecteurs ultrasonores de proximité est qu'ils ne sont pas sensibles à la

couleur des objets comme dans le cas des détecteurs d'infrarouge.

Pour notre système, nous avons opté pour les détecteurs ultrasonores de proximité.

### *1.6. Conclusion:*

Dans ce chapitre, nous avons défini l'étendu de notre projet; en résumé, les caractéristiques du robot sont :

- Robot mobile sur des roues, entraînées par des moteurs à courant continu.
- L'énergie est fournie par des batteries à plomb.
- Un détecteur ultrasonore pour éviter les obstacles équipe le robot, ainsi qu'une caméra capable de transmettre ses images par une communication sans fil au système de pilotage.
- Télécommandable par un système informatique de pilotage à travers une liaison hertzienne.
- Facilement adaptable à plusieurs types d'applications grâce à son équipement par divers détecteurs et capteurs.

# Chapitre 2 GÉNÉRALITÉS SUR LA ROBOTIQUE

## 2.1. Introduction:

Le terme "robot" a été introduit pour la première fois par l'auteur tchèque Karel Čapek, dans une pièce de théâtre appelée R.U.R. (Rossum's Universal Robots) et publiée en 1920. L'histoire parle d'une usine des hommes artificiels appelés Robots, qui sont autonomes et heureux de servir les humains.

D'après Karel Čapek, l'origine du mot est son frère Josef Čapek qui est aussi auteur et peintre. "Robot", qui est ensuite utilisé dans la plupart des langues slaves, signifie travail, travailleur, serf ou corvée.

Pourtant, il n'y a pas une définition commune de ce terme, mais un robot est généralement défini comme un dispositif mécanique capable d'exécuter des tâches humaines, ou pouvant se comporter comme l'homme.

Dès le début de la civilisation, l'homme a été fasciné par l'idée d'avoir une création qui lui ressemble et qui a le pouvoir et le désir de l'assister. Presque toutes les anciennes civilisations ont été impliquées dans l'esclavage, les esclaves sont utilisés pour les travaux durs et serviles.

Avant le début du premier millénium après J.C., l'homme a découvert la mécanique et les moyens qui lui permettaient de réaliser des machines complexes capables d'exécuter des tâches répétitives comme les pompes et les turbines à eau. La technologie de l'époque était primitive, les quelques machines qui ont été créées, exécutaient, malgré cela, des fonctions étonnement complexes.

Dans l'annexe 1, est présenté un bref historique sur les principales innovations dans ce domaine.

## 2.2. Classification des robots:

Actuellement, les robots peuvent être classés dans deux grandes catégories:

1. Les robots utilisés dans la Recherche comme les humanoïdes (ex : ASIMO) et comme les Nano robots.

2. Les robots commerciaux et industriels, capables d'exercer des tâches bien déterminées et d'une manière nettement plus précise et plus fiable que celle de l'être humain. De plus, ils peuvent être utilisés dans des milieux dangereux où les conditions sont inappropriées pour l'homme. Ces robots sont principalement utilisés dans les usines, le transport, l'exploration de l'espace, les opérations chirurgicales, l'armement, et les laboratoires de recherche.

Selon leurs activités, on peut distinguer les types de robots suivants :

- **Les robots multi-rôles :**

  Ces robots exécutent des multitâches d'une manière autonome; ils sont capables de naviguer

dans un milieu prédéfini, recharger eux-mêmes leurs batteries et pour améliorer leurs performances, ils peuvent s'interfacer avec d'autres systèmes, comme un système de contrôle d'accès, un ordinateur, un logiciel, un réseau, … etc.

Certains sont capables de reconnaître les personnes, de parler, de surveiller l'environnement et/ou de réagir à une alarme.

Ce type de robots, conçu pour imiter l'homme, est nommé "humanoïde".

- **Les robots utilisés dans des usines, telles que :**

  a. Production des voitures :

     Une usine des voitures contient typiquement des centaines des robots industriels totalement autonomes, dont leur rôle est principalement la soudure, le collage, la peinture et l'assemblage.

  b. L'emballage :

     Les robots industriels sont utilisés d'une manière intensive dans la palettisation et l'emballage des biens et des marchandises.

  c. L'électronique :

     Domaine où les robots sont utilisés exclusivement dans la fabrication des circuits imprimés. Ces robots sont capables de mettre en place des centaines des milliers des composants électroniques par heure, d'une façon plus rapide, fiable et précise que l'homme.

  d. Les véhicules auto-guidés:

     Ce sont des robots mobiles qui suivent des marques et des lignes tracées sur les trajets à suivre, ou qui utilisent la vision ou la lumière laser pour la navigation. Ils sont utilisés pour transporter des biens ou des marchandises dans des grands magasins et des larges dépôts.

- **Les robots utilisés dans les milieux dangereux, inaccessibles et hasardés :**

Certains travaux, que l'être humain ne peut pas (ou préfère ne pas) effectuer parce que leurs conditions de réalisation sont dangereuses, lointaines et/ou inaccessibles, peuvent être exécutés grâce à des robots conçus spécialement pour ce genre d'activités. Par exemple, les travaux dangereux (exploration d'un volcan), ou physiquement inaccessibles (exploration des planètes lointaines ou nettoyage dans un long tuyau) constituent des domaines d'applications des Télé-robots (ou robots télé-contrôlés par un opérateur humain).

Ces robots sont aussi utilisés dans les domaines militaire (pour désamorcer les bombes, [figure 2.1)], spatial (exploration de l'espace tel que le robot Pathfinder utilisé pour l'exploration de la planète Mars, [figure 2.2]), et chirurgical (figure 2.3).

**Figure 2.1**, Robot utilisé pour désamorcer les bombes en Iraq.

- **Les robots domestiques :**

  Ces robots exécutent des tâches simples mais non désirées par l'être humain, telles que les tâches ménagères comme le nettoyage, aspirateur ou lavage. La Figure 2.4 montre un robot aspirateur de nettoyage.

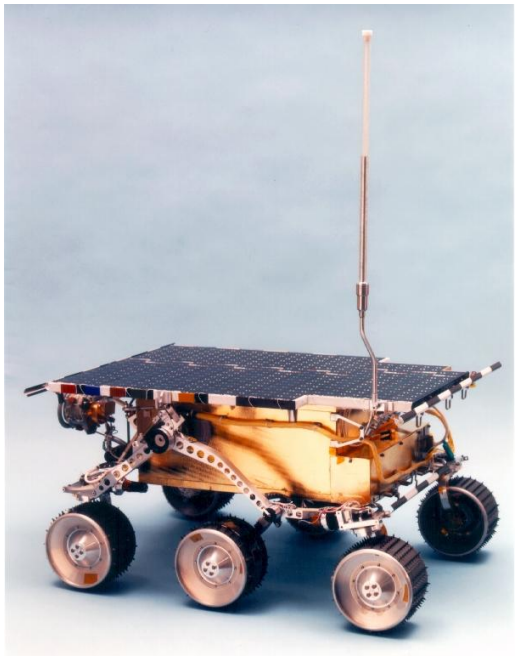


**Figure 2.2**, Robot PathFinder utilisé pour l'exploration de la planète Mars.

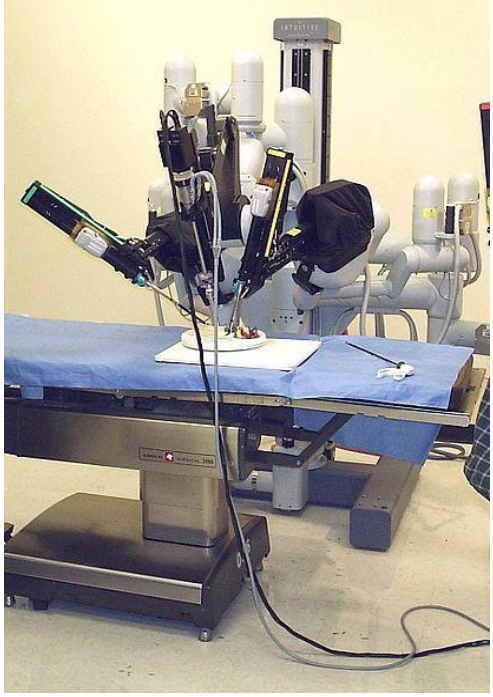


**Figure 2.3**, Robot laparoscopie utilisé dans les opérations chirurgicales.


**Figure 2.4**, Robot aspirateur de nettoyage.

- **Les robots de recherche :**

Autre que les robots humanoïdes et les robots industriels, les laboratoires de recherche spécialisés se concentrent à développer des nouveaux types de robots, avec des nouvelles méthodes pour les construire. Parmi lesquels, on peut citer:

- Les Nano robots:

Cette technologie est actuellement hypothétique, le but est de créer des robots à l'échelle de nanomètre. Jusqu'à maintenant les chercheurs ont construit quelques parties de ces robots, comme les capteurs et les moteurs moléculaires. Les chercheurs espèrent qu'ils créeront un robot de mêmes dimensions qu'un virus ou qu'une bactérie.

- Les robots reconfigurables :

Quelques laboratoires ont investi dans la fabrication des robots qui changent leurs formes

physiques. Cette technique, loin d'être inspirée des émissions des sciences-fictions, cherche à construire les robots de formes et fonctionnalités différentes en rassemblant des éléments cubiques de base l'un à côté de l'autre (figure 2.5).

- Les robots mous :

Ils sont fabriqués à partir des matériaux à base du silicone et munis des actionneurs spécifiques de manière les rendant malléables avec des systèmes de contrôle par logique floue ou par un réseau neuronal.

- Essaim des robots :

Ce type des robots est inspiré par les colonies des insectes; ce sont des petits robots qui exécutent ensemble la même tâche d'une manière coopérative et efficace. Ils peuvent trouver une chose ou une personne cachée ou perdue, nettoyer ou même espionner.

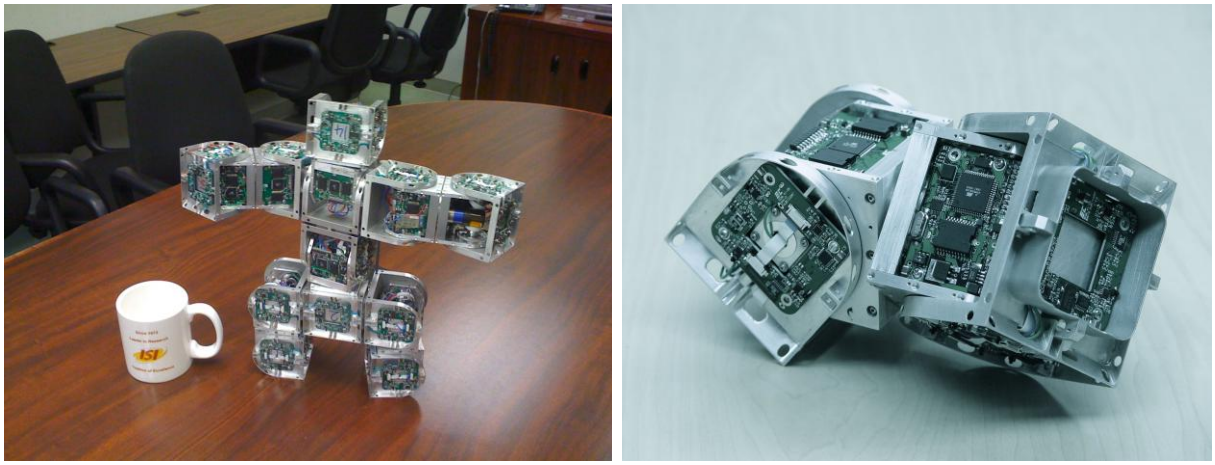


**Figure 2.5,** SuperBot: la figure à droite montre un cube qui constitue l'élément de base pour la réalisation du robot. La figure à gauche montre un assemblage possible des cubes élémentaires.

## *2.3. Conclusion:*

La robotique, comme toutes les technologies, a commencé petite. Mais le progrès dans le domaine des robots a été douloureusement lent. La robotique est une industrie neuve, même en tenant compte des automates spéciaux largement utilisés dans les fabrications et les usines.

Que signifie cela pour la robotique expérimentatrice ?

Il signifie qu'il y a encore de la place pour la croissance et l'innovation dans ce domaine, encore plus que dans beaucoup bien d'autres disciplines de la haute technologie.

Il est vrai qu'au Liban les supports et les fondements de la recherche dans ce domaine sont très limités et parfois inexistants, mais je pense qu'il y a toujours lieu pour innover pour des applications particulières dans le domaine des robots mobiles autoguidés, des robots télé-contrôlés en particulier et des robots industriels, en général.

# Chapitre 3 : LES RÉSEAUX DE NEURONES

### *3.1. Qu´est-ce que le calcul neuronal :*

Le calcul neuronal est utilisé pour le traitement des informations. Contrairement au traitement traditionnel, le calcul neuronal nécessite un processus d´apprentissage à l´aide d´un réseau de neurones artificiels, architecture qui adapte sa sortie à son entrée selon une règle d´apprentissage. Ensuite ce réseau formé par les données est utilisé pour une tâche dans une application particulière.

Le réseau de neurone a la capacité de s´adapter à son environnement d´une manière interactive semblable à son homologue biologique.

Cette perspective est passionnante à cause des énormes possibilités qui existent pour réaliser certaines fonctions avec un réseau de neurones artificiels qui imite (à un certain degré) son comparable biologique.

Quoique les gens ne soient pas rapides ou précis autant qu´ordinateur, ils sont en général bien meilleurs pour percevoir et identifier un objet d'intérêt dans une scène normale ou pour interpréter le langage naturel.

Pourquoi pouvons nous exécuter certaines fonctions bien mieux qu'un ordinateur? La réponse à cette question jusqu´à présent n'est pas totalement comprise, pourtant, une raison pour laquelle nous sommes bien meilleurs à la reconnaissance des objets dans une scène complexe est due à la manière dont notre cerveau est organisé. Ce dernier utilise une architecture "informatique" bien orchestrée pour résoudre des problèmes très complexes : l'unité de traitement de base dans le cerveau est le neurone.

Les neurones sont plus lents que la porte logique de silicium. Cependant, le cerveau peut compenser la vitesse opérationnelle relativement lente du neurone par le traitement des données dans une architecture parallèle très dense qui est massivement interconnectée.

Il est estimé qu´un cerveau humain doit contenir de l´ordre de $10^{11}$ neurones et, à peu près, trois fois ce nombre de connections ou synapses. Par conséquent, le cerveau est un ordinateur parallèle, adaptif et non linéaire qui est capable d´organiser les neurones pour réaliser certaines tâches. Le système du calcul neuronal modélise très approximativement le cerveau d'humain.

Par exemple, les capacités d'apprendre et de généraliser sont les principales caractéristiques d´un réseau de neurones artificiels. Ce dernier est entraîné par la présentation de plusieurs modèles qu' il doit apprendre selon une règle d´apprentissage (Algorithme).

Un réseau de neurones emmagasine la connaissance acquise durant l´apprentissage dans les poids synaptiques des neurones. Comment le réseau de neurones est-il organisé ? Quelles sont les règles d´apprentissage utilisées pour ajuster les poids synaptiques ? Quel est le critère qui dicte la fin de l´apprentissage? Les réponses à ces questions caractérisent le type de réseau.

Le domaine d'application des réseaux neuronaux est devenu, au fil des années, très vaste et interdisciplinaire ; actuellement, il est utilisé par des chercheurs dans différents domaines d'ingénierie, mais aussi en mathématiques, et en économie.

Le réseau neuronal artificiel offre une approche de calcul neuronal permettant de résoudre des problèmes complexes pour lesquels les solutions n'étaient pas facilement abordables. Les applications des réseaux de neurones incluent (sans qu'elles soient exhaustives) la prédiction et la provision, la mémorisation, les techniques d´approximation, la compression des informations, la reconnaissance et la synthèse de la parole, la modélisation des systèmes non linéaires, le contrôle non linéaire, la classification des données, l'extraction des caractéristiques, la résolution des problèmes algébrique, la résolution des équations différentielles, … etc.

Par rapport aux approches classiques, l´utilisation du calcul neuronal pour résoudre ce type de problèmes offre des avantages indéniables dont, par exemple, la tolérance aux défauts: Si, dans une implantation matérielle d'un réseau, un neurone ou l'une de ses connections sont endommagés, la performance de l'ensemble du réseau serait légèrement affectée.

Cette propriété contribue à la nature robuste du réseau de neurone due principalement au fait que les informations sont réparties dans le réseau.

Les neurones artificiels sont non linéaires donc le réseau lui-même est non linéaire. Cette non linéarité est très importante surtout si le processus physique à modéliser est non linéaire.

De plus, la nature adaptative du réseau est aussi importante, l´adaptation des poids synaptiques se fait en fonction des différentes situations présentées au réseau; ce qui nous permet d'avoir un réseau dynamique qui s'adapte, continuellement et en temps réel, ce qui constitue un atout évident surtout dans un environnement non stationnaire.

### *3.2. Le calcul neuronal et la science neuronale:*

Certains développements dans ce domaine ont été inspirés par les recherches de la science neuronale. Pour cela il est approprié de présenter une vue générale sur le réseau de neurone biologique.

Le système nerveux, dont le cerveau constitue l´élément central, est un réseau de neurones très vaste et complexe. Les 100 milliards de neurones qui constituent le cerveau sont interconnectés à travers des sous-réseaux appelés nucleus. Chaque nucleus a une fonction définie qui consiste à trier et traiter les informations reçues avant de les envoyer vers d'autres sous-réseaux (nucleus) ; le résultat de ce traitement collectif sera délivré sous forme d'une commande transmise pour lancer une action.

Les capteurs et les nucleus sont impliqués dans un processus complexe de décomposition de l´information captée en des composants fondamentaux constituant les principales caractéristiques des organes des sens.

Ces décompositions sont distinctes pour chaque sens; par exemple, l´œil et le cerveau décomposent l´image vue suivant la couleur, l´intensité, la direction, les mouvements, l´échelle, … etc. Après cette analyse, les composants sont transmis vers d´autres sous-réseaux pour l´évaluation sélective et la

reconstruction partielle.

Le neurone biologique a servi comme inspiration pour le neurone artificiel, il est important de s´adresser au système biologique pour modeler ainsi les neurones artificiels plus efficacement. Le cerveau et les autres parties du système nerveux se composent de plusieurs types de neurones, qui diffèrent par leurs propriétés électriques, nombres, dimensions et architectures de connectivité, (voir la figure 3.1 et le tableau 3.1 : Exemples d´un Parvo (petit) et d´un Magno (large) cellules).

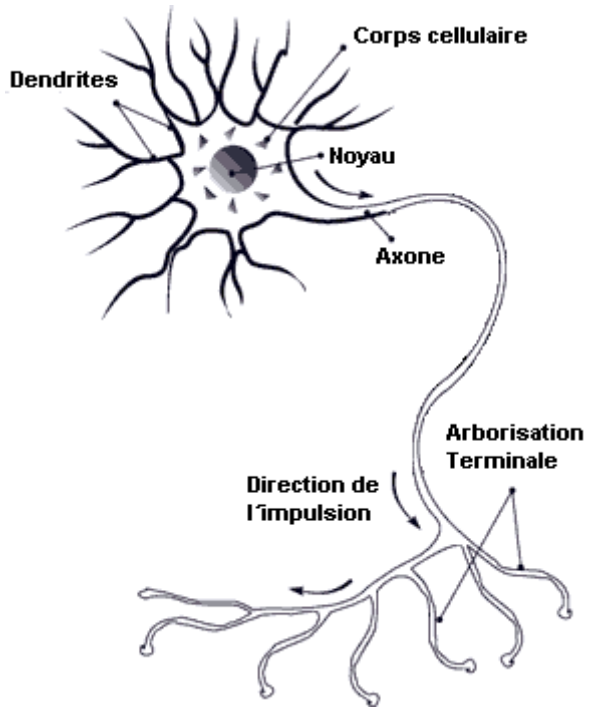



**Figure 3.1,** Schéma d´une cellule nerveuse (neurone).

Le neurone biologique est constitué de trois composants principaux : Dendrites, corps cellulaire et axone. Les dendrites reçoivent les signaux d´autres neurones; les axones sont connectés aux dendrites d´autres neurones et au corps cellulaire par des connecteurs appelés synapses (voir figure 3.2). Selon le type neuronal, le nombre des connecteurs synaptiques peut aller de quelques centaines à 10000. À cause des propriétés électriques des membranes neuronales, les signaux qui arrivent à la dendrite décroissent rapidement dans le temps et sur la distance, perdant ainsi la capacité de stimuler le neurone sauf s´ils sont renforcés par d'autres signaux qui se produisent presque dans le même temps et/ou dans une localisation tout proche.

Le corps cellulaire (soma) additionne les signaux venant des dendrites et les signaux venant des synapses à sa surface, si ces signaux reçus sont suffisants (atteignant certain seuil) pour stimuler le neurone, ce dernier produit une action et transmet cette action par son axone vers d´autres neurones ou vers des cellules à l'extérieur du système nerveux comme les muscles. D´ailleurs, l´intensité d´une entrée dépend du nombre d´actions générées par seconde non pas de sa dimension. Exemple, la dimension et la forme d´une forte entrée sont les mêmes que celles d´une faible entrée, cependant, une entrée forte produit davantage d´actions par seconde que l´entrée faible.

**Tableau 3.1, Différences anatomiques et physiologiques entre les cellules Parvo et Magno ganglion avec quelques conséquences des possibles comportements.**

| | Cellule *Parvo ganglion* | Cellule *Magno ganglion* |
|---|---|---|
| Différences anatomiques | Corps cellulaire petit<br>Branchement très dense<br>Branchement court<br>Majorité des cellules | Corps cellulaire large<br>Branchement non dense<br>Branchement long<br>Minorité des cellules |
| Différences physiologiques | Taux de conduction lent<br>Réponse continue<br>Petite zone de réception<br>Sensitivité de petit contraste<br>Sensitivité à la couleur | Taux de conduction rapide<br>Réponse transitoire<br>Large zone de réception<br>Sensitivité de grand contraste<br>Aveugle à la couleur |
| Conséquences des possibles comportements | Analyse des formes détaillées<br>Analyse spatiale<br>Vision de la couleur | Détection de mouvement<br>Analyse temporelle<br>Perception plus profonde |

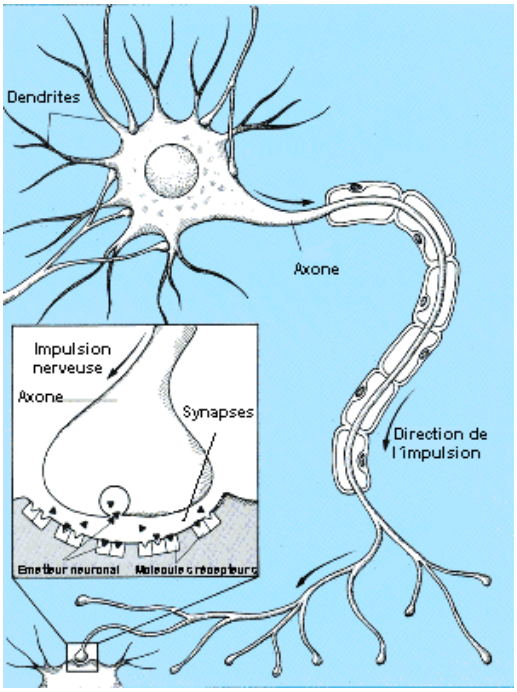



**Figure 3.2,** Schéma d´une structure d´un neurone biologique.

Les synapses sont des points de contact qui connectent les terminaux de l´axone à leurs cibles. Ses structures spéciales reliant par exemple les axones aux dendrites, aux corps cellulaires, aux terminaux nerveux, aux muscles, ou aux glandes, sont caractérisées par des structures chimiques et électriques. La synapse est constituée de trois éléments primaires: Le terminal du nerf, la fissure de synapse ou intervalle, la membrane post-synaptique, (voir figure 3.3).

Quand une impulsion arrive au terminal du nerf, ce dernier convertit le signal électrique en un signal chimique par une série de réactions chimiques, dans l´étape finale de cette conversion le nerf libère une

substance appelée émetteur neuronal. Celui-ci agit sur la membrane post-synaptique; le système nerveux libère plusieurs types d'émetteurs neuronaux, par contre, un terminal du nerf libère seulement un seul type d´émetteur neuronal. L´émetteur neuronal se diffuse à travers la fissure de synapse qui est un espace relativement vide pendant environ 2 ms. Chaque émetteur neuronal est relié à un récepteur encastré dans la membrane post-synaptique, quand il reçoit l´émetteur neuronal il déclenche alors une réponse électrique et biochimique qui conduit à un changement dans la potentielle de la membrane post-synaptique. Il y a deux classes majeures d´émetteur neuronal : excitateur et inhibiteur. Les premiers dépolarisent la membrane, mais une seule synapse est très petite pour générer une action potentielle tout seule, quand elle est additionnée à d´autres dépolarisations concurrentes dans des centaines de synapses elles peuvent générer une action potentielle. Les inhibiteurs cause un effet opposé et "hyperpolarise" la membrane post-synaptique ainsi ils invalident les actions des excitateurs et dans des certains cas ils empêchent la régénération d´une action potentielle. Donc une action représente une sommation des excitateurs dans le neurone, à son tour cette action potentielle voyage à la fin de l´axone où le neurone communique avec les autres par les synapses.

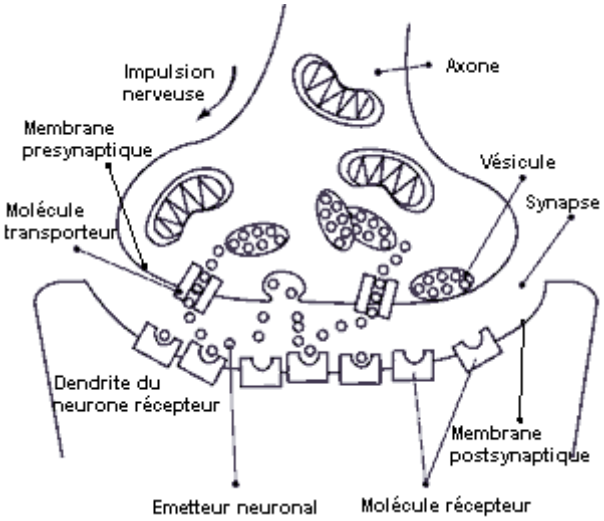



**Figure 3.3,** Dessin détaillé d´une synapse.

Comme un exemple de similarité entre un réseau de neurone artificiel et biologique, le tableau 3.2 compare trois différentes architectures de réseaux.

**Tableau 3.2 Exemple des architectures des réseaux de neurones et leurs homologues de systèmes nerveux.**

| **Type de Réseau de neurones** | **Système nerveux** |
|---|---|
| Réseau alimentation directe couche unique. | Rare excepté dans les voies les plus simples de réflexe. |
| Réseau alimentation directe multi couches. | Commun et complexe, typiquement ayant plusieurs niveaux de couches cachées. Celles partiellement reliées sont plus fréquentes que celles entièrement reliées. |
| Réseau récurrent. | Contre-réaction négative: il est un peu plus compliqué que son analogue artificiel. Essentiel et dominant dans le système nerveux. |

### *3.3. Le modèle de base d´un neurone artificiel :*

Jusqu´à présent on a vu qu´un réseau de neurones artificiels (RNA) est formé par plusieurs neurones artificiels. Un neurone artificiel est appelé aussi élément de traitement, un nœud ou une unité logique avec un seuil. Sur la Figure 3.4 représentant le modèle d´un neurone artificiel, on y trouve quatre parties principales :

(1) Une série de synapses auxquelles on associe des poids synaptiques, la figure 3.4 montre que l´entrée vers les synapses est un vecteur $x \in \Re^{n\times 1}$ tel que $x = [x_1, x_2, \ldots, x_n]^T$ ou $x_j$ sont les composants du vecteur avec $j = 1,2,\ldots,n$. Le composant $x_j$ est l´entrée de la $j^{ème}$ synapse et il est connecté au neurone $q$ à travers le poids synaptique $w_{qj}$ ( $x_j$ est alors multiplié par $w_{qj}$ ).

(2) Le nœud de sommation additionne tous les signaux qui lui sont envoyés; cette opération conduit à une sortie $u_q$ qui est une combinaison linéaire de toutes les entrées des synapses.

(3) La fonction d´activation ou la fonction d´écrasement $f(\bullet)$; quand elle est non-linéaire elle sert à limiter la sortie du neurone $y_q$. Elle peut être binaire ou bipolaire ou, dans certains cas, linéaire. quand elle est non linéaire la sortie est typiquement limitée entre $[0,1]$ (binaire) ou $[-1,1]$ (bipolaire).

(4) le seuil $\theta_q$ est typiquement appliqué de l´extérieur pour diminuer l´entrée cumulative de la fonction d´activation, donc, $\theta_q$ est soustrait de la sortie linéaire $u_q$ avant d´être appliquée à la fonction d´activation; $\theta_q$ peut être aussi un facteur de forçage (il force l´activation de la fonction) dans ce cas, il est additionné à la sortie $u_q$. La relation entre l´entrée $v_q$ de $f(\bullet)$ et la sortie linéaire $u_q$ sera :

$$v_q = u_q - \theta_q \tag{3.1}$$

Mathématiquement, on peut décrire l´opération du neurone artificiel dans figure 3.4 par les équations suivantes:

La sortie de l´additionneur linéaire est donnée par : $$u_q = \sum_{j=1}^{n} w_{qj} x_j = w_q^T x = x^T w_q \tag{3.2}$$

avec $x$ présenté ci-dessus, $w_q = [w_{q1}, w_{q2}, \ldots, w_{qn}]^T \in \Re^{n\times 1}$.

La sortie de la fonction d´activation est : $$y_q = f(v_q) = f(u_q - \theta_q) \tag{3.3}$$

Donc, en utilisant les équations (3.2) et (3.3) la sortie du neurone est donnée par :

$$y_q = f\left(\sum_{j=1}^{n} w_{qj} x_j - \theta_q\right) \tag{3.4}$$

La figure 3.5 montre un autre modèle du neurone artificiel, dans ce modèle le seuil est incorporé dans

les poids synaptiques, le vecteur $w_q$ pour le neurone q et le vecteur d´entrée est augmenté par $x_0$ , donc, $x \in \Re^{n+1\times 1}$ et $w_q \in \Re^{n+1\times 1}$ .

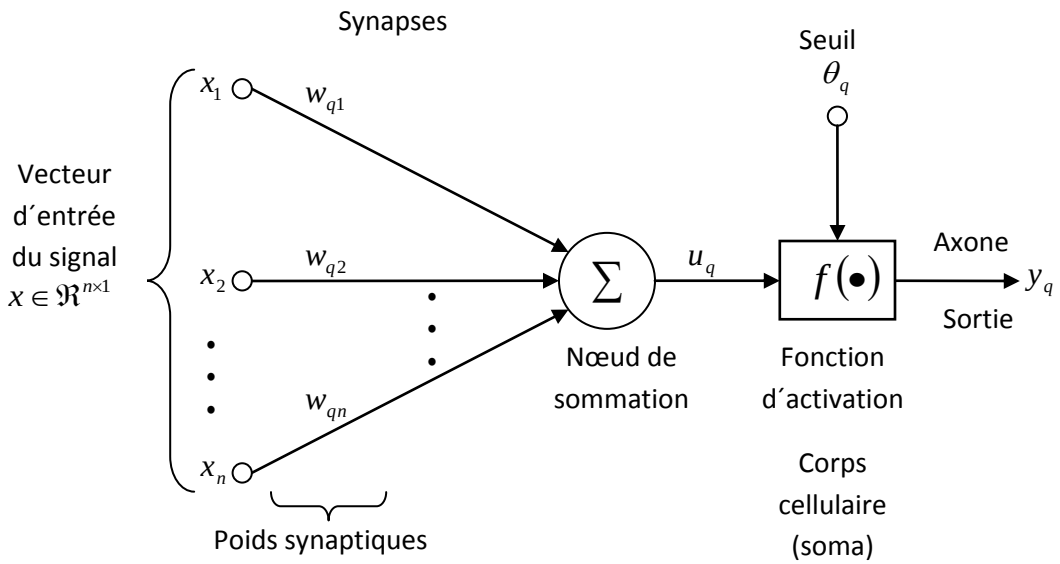



**Figure 3.4** Un modèle non linéaire d´un neurone artificiel.

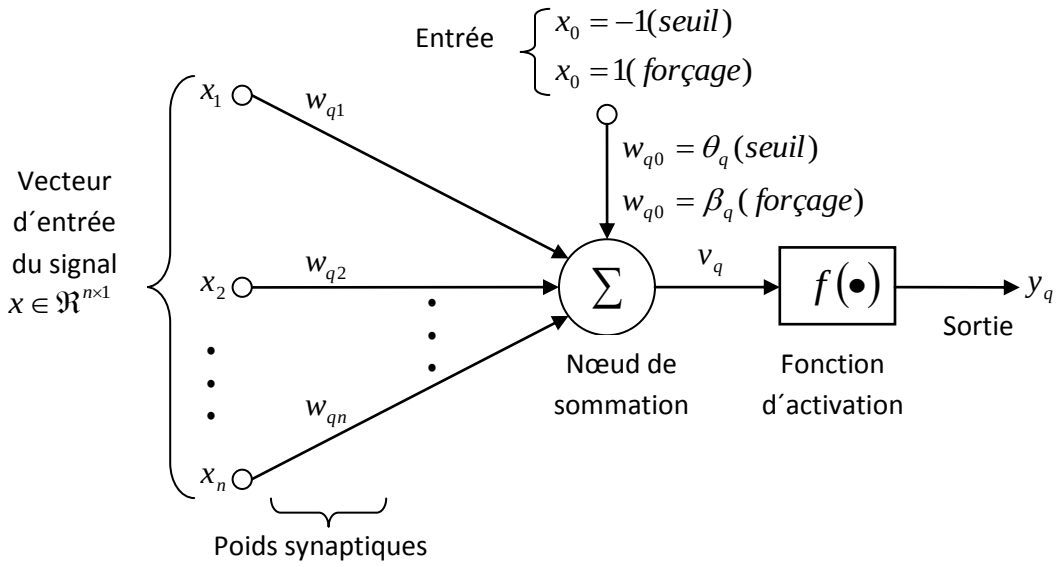



**Figure 3.5** Un autre modèle non linéaire d´un neurone artificiel.

Le potentiel d´activation effectif s'écrit alors : $v_q = \sum_{j=0}^{n} w_{qj} x_j$ (3.5)

Et la sortie de neurone q est donnée par: $y_q = f(v_q)$ (3.6)

### 3.4. Les fonctions d´activation :

La fonction d´activation ou fonction de transfert peut être linéaire ou non linéaire, il y a différents types de fonctions d´activation, la sélection d´un type dépend du problème à résoudre par le réseau de neurones. On représente ici les quatre fonctions les plus utilisées, notre référence sera la figure 3.5.

- Le premier type est la fonction linéaire dont les valeurs sont continues; mathématiquement la sortie de la fonction d´activation du neurone $q$ est

$$y_q = f_{lin}(v_q) = v_q \tag{3.7}$$

Avec $v_q$ la sortie de l´additionneur et l´entrée de la fonction d´activation comme le montre la figure 3.5.La fonction linéaire est représentée par la courbe de la figure 3.6.

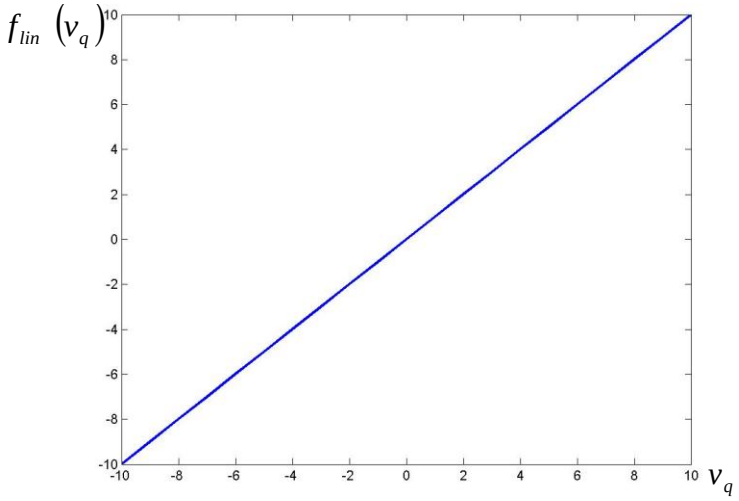



**Figure 3.6** Fonction d´activation linéaire.

- Le second type correspond à la fonction est l´échelon; cette fonction peut être binaire (figure 3.7) ou bipolaire (figure 3.8). La sortie de la fonction échelon binaire peut être écrite comme suit:

$$y_q = f_{ech}(v_q) = \begin{cases} 0 & \text{si } v_q < 0 \\ 1 & \text{si } v_q \geq 0 \end{cases} \tag{3.8}$$

Et la sortie de la fonction échelon bipolaire ou échelon symétrique s'écrit ainsi :

$$y_q = f_{echs}(v_q) = \begin{cases} -1 & \text{si } v_q < 0 \\ 0 & \text{si } v_q = 0 \\ 1 & \text{si } v_q > 0 \end{cases} \tag{3.9}$$

Cette fonction est nommée parfois *signum,* $f(\bullet) = \text{sgn}(\bullet)$.

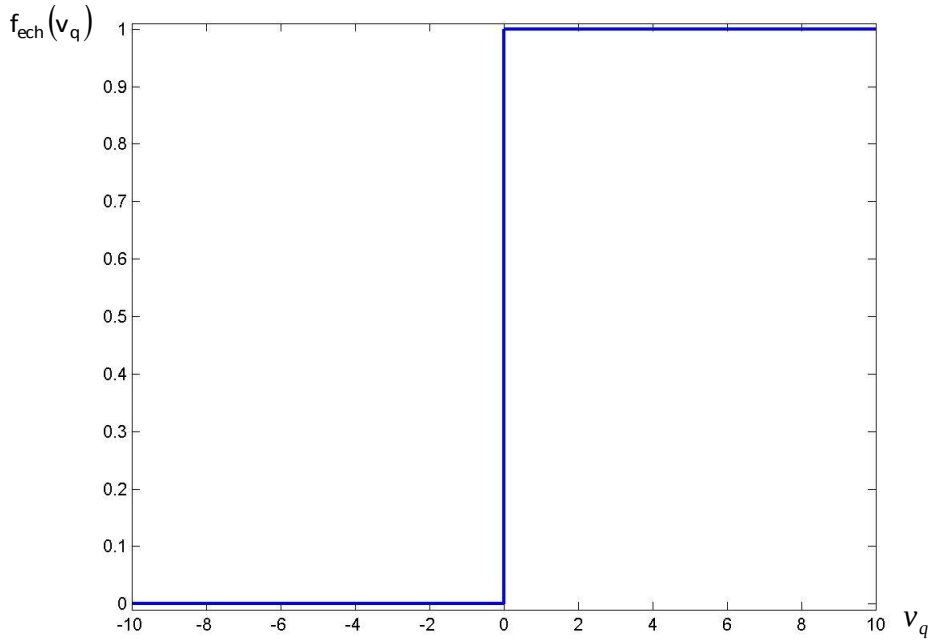



**Figure 3.7** Fonction d´activation échelon .

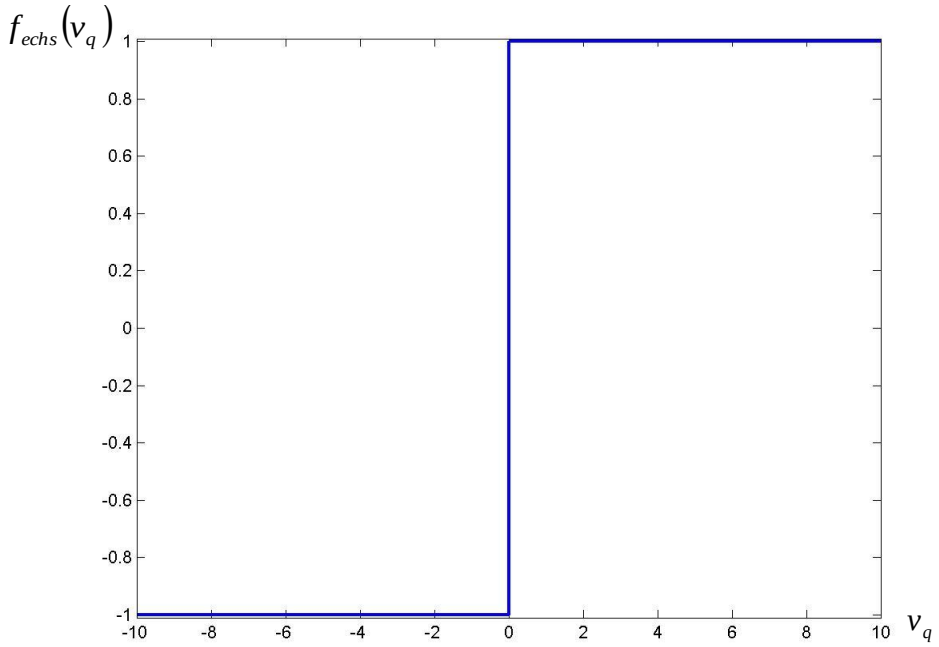



**Figure 3.8** Fonction d´activation échelon symétrique.

- Le troisième type est la fonction linéaire à saturation ou linéaire limitée; cette fonction peut être aussi binaire (figure 3.9) ou bipolaire (figure 3.10), la fonction bipolaire est référencée aussi par linéaire limitée symétrique. La fonction linéaire à saturation binaire est donnée par la fonction :

$$y_q = f_{lins}(v_q) = \begin{cases} 0 & \text{si } v_q < -\frac{1}{2} \\ v_q + \frac{1}{2} & \text{si } -\frac{1}{2} \le v_q \le \frac{1}{2} \\ 1 & \text{si } v_q > \frac{1}{2} \end{cases} \tag{3.10}$$

et celle à saturation bipolaire par : $y_q = f_{linss}(v_q) = \begin{cases} -1 & \text{si } v_q < -1 \\ v_q & \text{si } -1 \le v_q \le 1 \\ 1 & \text{si } v_q > 1 \end{cases}$ (3.11)

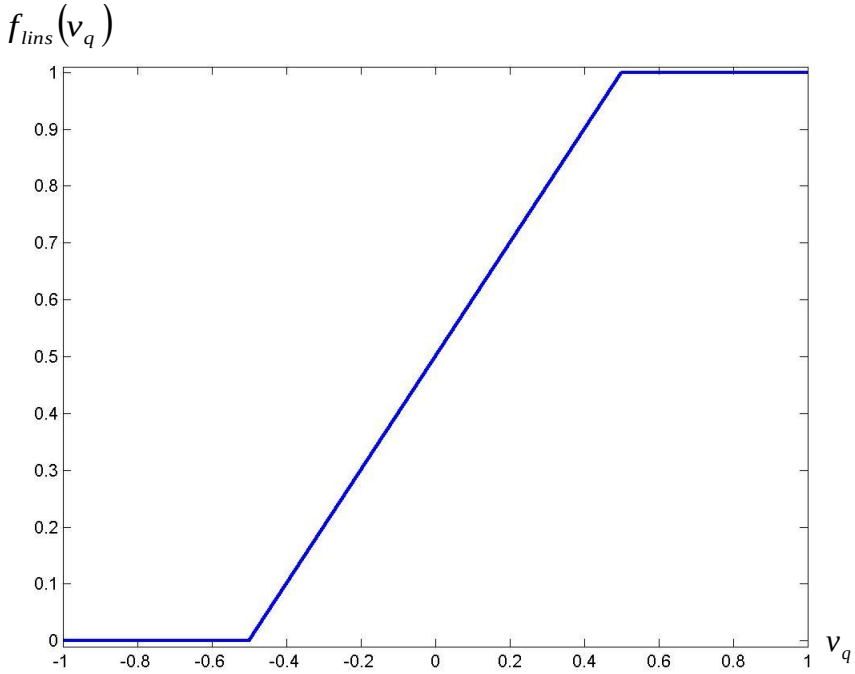



**Figure 3.9** Fonction d´activation linéaire à saturation binaire.

- La quatrième fonction est connue par *sigmoïde* et ici on en représente deux types; cette fonction non linéaire est la plus utilisée pour construire un réseau de neurone artificiel, le premier type est la fonction *sigmoïde* binaire (figure 3.11), elle est donnée par:

$$y_q = f_{bs}(v_q) = \frac{1}{1 + e^{-\alpha v_q}} \tag{3.12}$$

où $\alpha$ représente la pente de la fonction. En changeant ce paramètre, différentes formes de la fonction peuvent être obtenues comme le montre la figure 3.11 pour différentes valeurs de $\alpha$ .Contrairement à la fonction d´activation échelon, la fonction *sigmoïde* est continue et dérivable à l´origine; la dérivabilité de la fonction d´activation joue un rôle très important dans le calcul neuronal. La dérivée de la fonction d´activation *sigmoïde* binaire en fonction de la sortie du l´additionneur $v_q$ est donnée par :

$$g_{bs}(v_q) = \frac{df_{bs}(v_q)}{dv_q} = \frac{\alpha e^{-\alpha v_q}}{\left(1+e^{-\alpha v_q}\right)^2} = \alpha f_{bs}(v_q)\left[1 - f_{bs}(v_q)\right] \tag{3.13}$$

À l´origine la pente du *sigmoïde* binaire est $\alpha/4$, donc, quand on augmente $\alpha$ on se rapproche de la fonction échelon; la figure 3.12 montre les dérivées du *sigmoïde* binaire pour $\alpha = 1$ et $\alpha = 0.5$.

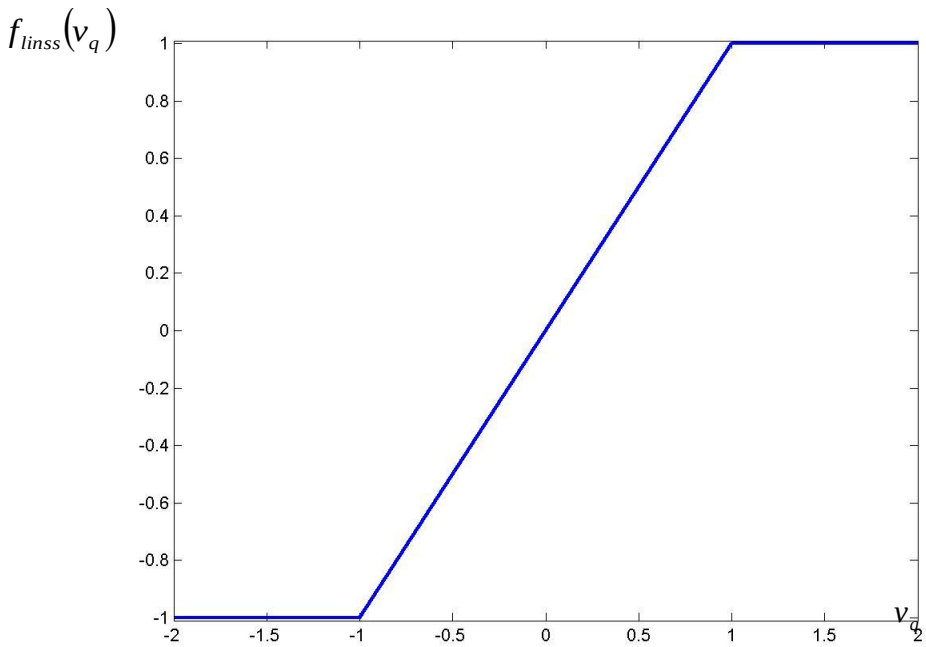



**Figure 3.10** Fonction d´activation linéaire à saturation bipolaire.

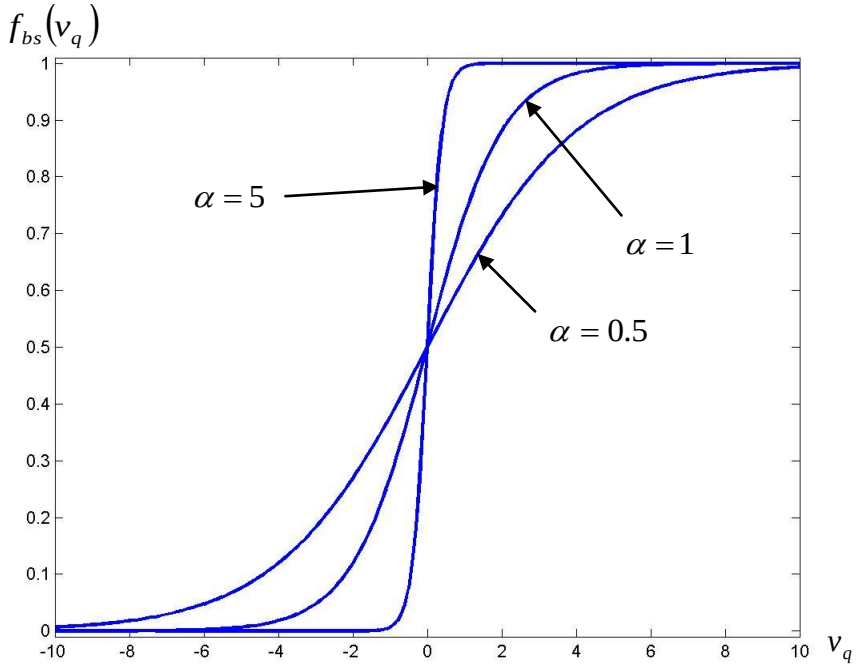



**Figure 3.11** Fonction d´activation *sigmoïde* binaire.

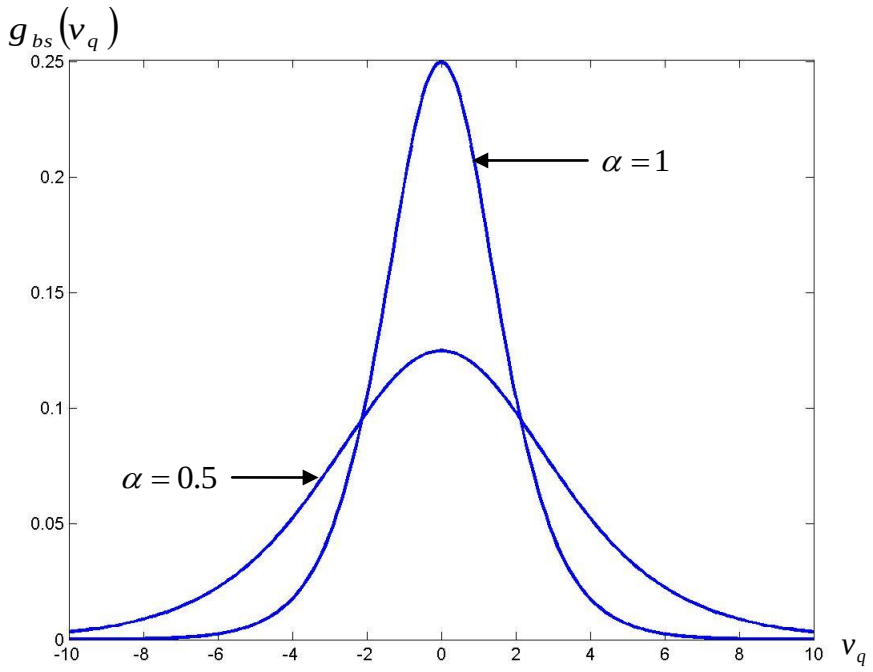



**Figure 3.12** Dérivée de la fonction d´activation *sigmoïde* binaire.

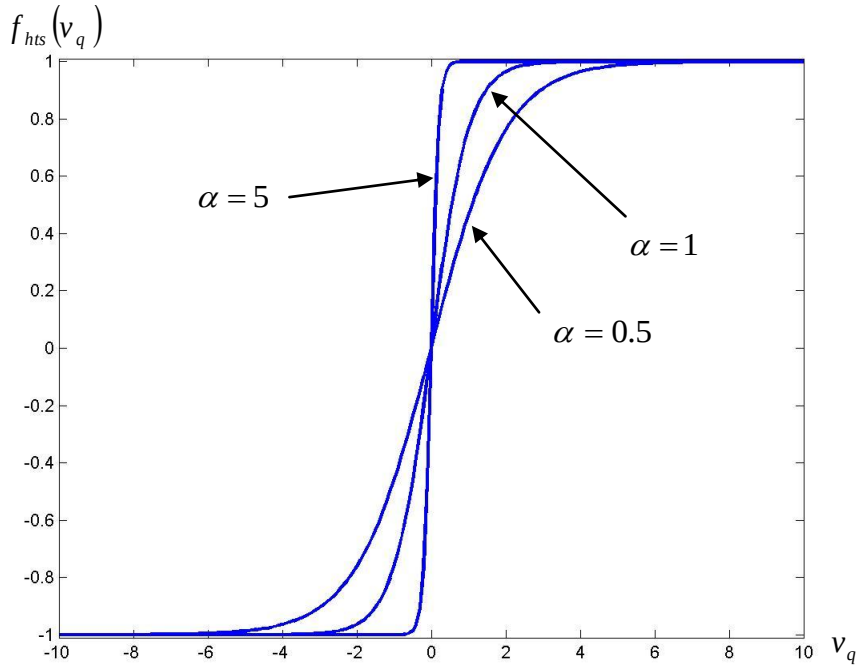



**Figure 3.13** Fonction d´activation *sigmoïde tangente hyperbolique*.

Le deuxième type est la version bipolaire ou la fonction *sigmoïde tangente hyperbolique* (figure 3.13); cette fonction est donnée par:

$$y_q = f_{hts}(v_q) = \tanh(\alpha v_q) = \frac{e^{\alpha v_q} - e^{-\alpha v_q}}{e^{\alpha v_q} + e^{-\alpha v_q}} = \frac{1 - e^{-2\alpha v_q}}{1 + e^{-2\alpha v_q}} \tag{3.14}$$

et sa dérivée par: $$g_{hts}(v_q) = \frac{df_{hts}(v_q)}{dv_q} = \alpha\left[1 + f_{bs}(v_q)\right]\left[1 - f_{bs}(v_q)\right] \quad (3.15)$$

Ce qui est intéressant dans cette fonction est que sa dérivée est exprimée par la fonction elle-même, et cela devient important quand on développe une règle d´apprentissage.

### *3.5. Le modèle Hopfield d´un réseau de neurone :*

Le modèle de base est donné par la figure 3.14 avec une fonction d´activation échelon binaire, bipolaire, ou *sigmoïde tangente hyperbolique*. Les réseaux Hopfield sont des réseaux récurrents, caractérisés par leur contre-réaction.

La sortie de neurones avant le retard $z^{-1}$ est donnée par :

$$y_q(k+1) = f_{echs}\left[v_q(k+1)\right] \quad (3.16)$$

avec $$v_q(k+1) = \sum_{j=1}^{n} w_{qj} x_j(k) - \theta_q \quad (3.17)$$

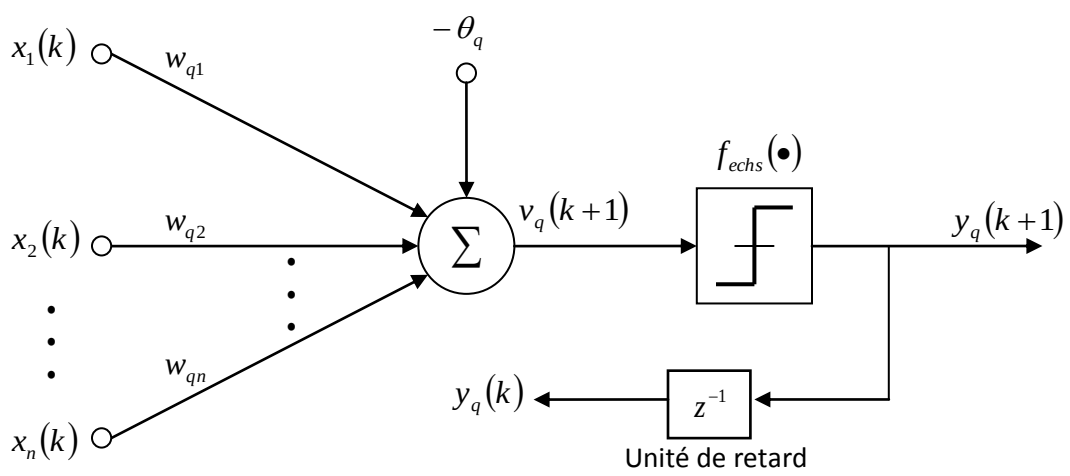



**Figure 3.14** Le modèle Hopfield d´un neurone artificiel.

La sortie devient alors : $$y_q(k+1) = f_{echs}\left[\sum_{j=1}^{n} w_{qj} x_j(k) - \theta_q\right] \quad (3.18)$$

L´unité de retard $z^{-1}$ retarde la sortie de la fonction d´activation d'une période ce qui donne, à l'instant k+1, la sortie $y_q(k)$. Pour le réseau de Hopfield cette quantité est injectée à l´entrée des neurones à l'exception de l´entrée du neurone responsable de cette sortie.

# PARTIE II

## Chapitre 4 PARTIE MECANIQUE

## Chapitre 5 PARTIE ELECTRONIQUE

## Chapitre 6 PARTIE LOGICIELLE

# Chapitre 4 PARTIE MÉCANIQUE

## 4.1. Introduction:

Le travail individuel pour construire un robot ne peut pas prétendre aux meilleures performances à tous les niveaux. Pour simplifier la tâche, on doit renoncer à certains aspects liés plutôt à la forme et aux apparences, qu'à l'essentiel qui, pour quelqu'un qui souhaite innover, reste primordial.

Ainsi, nous avons choisi d'utiliser un matériel standard facilement accessible dans notre entourage ou sur le marché local pour construire le châssis de notre robot mobile.

Dans ce chapitre, nous décrivons, étape par étape, comment nous avons procédé à la réalisation mécanique d'un robot mobile sur roues d'une manière simple, utile et pas chère.

## 4.2 Le châssis du robot :

Pour le châssis du robot, nous avons choisi d'utiliser, tout simplement, une boîte de jonctions électriques, de type PVC dont les dimensions sont 400x300x80 $mm^3$ (figure 4.1 et figure 4.2).

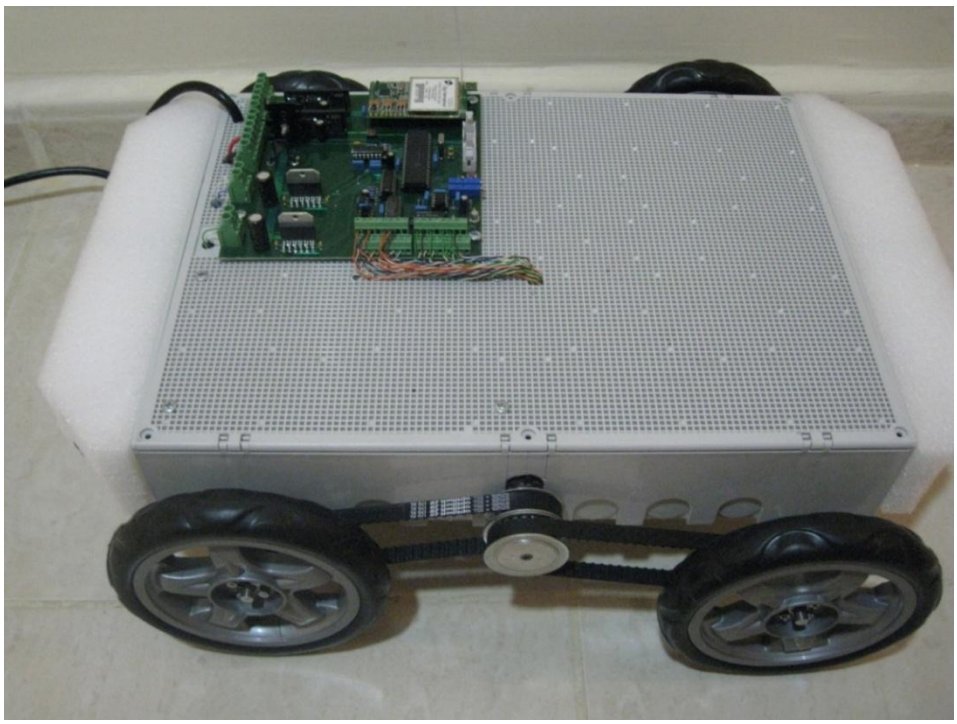


**Figure 4.1**, Vue générale du robot.

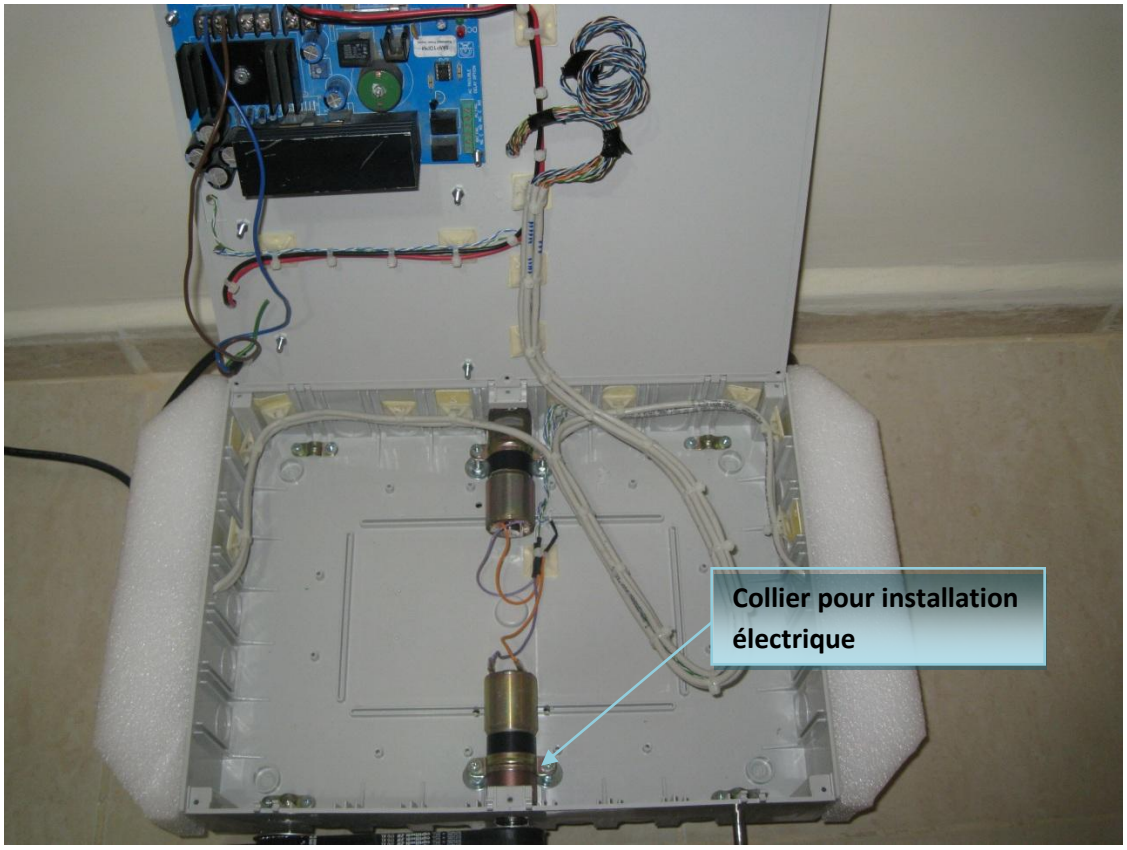



**Figure 4.2**, Vue de l'intérieur du robot.

En effet, ce type de boîtes en PVC sont disponibles à très bas prix, et en dimensions diverses.

De plus, le matériau en PVC est un bon choix, vu ses caractéristiques intéressantes pour notre application, telles que, légèreté, malléabilité et rigidité.

Si l'application exigeait un robot lourd, le PVC ne serait plus le meilleur choix.

## *4.3 Choix et fixation des roues:*

Pour rendre l'engin mobile, nous l'avons équipé par des roues entraînées par des moteurs électriques.

Les caractéristiques des moteurs à choisir dépendent de celles des roues, mais aussi du poids maximal qu'il faut entraîner, de la vitesse maximale attendue et de l'accélération maximale prévue.

Nous avons commencé par définir les paramètres de mouvement de notre robot comme suit:

- Vitesse maximale de déplacement de 0,5 m/s.
- Accélération maximale de 0,5 $m/s^2$.
- Le poids maximal à entraîner correspond à une masse de 10 kg.
- Les roues ont 15 cm de diamètre.

À partir de ces hypothèses, nous calculons, dans le paragraphe suivant, les valeurs caractéristiques des 2 moteurs qui vont équiper notre engin.

Les 4 roues, de 15 cm de diamètre, que nous avons utilisées étaient celles d'un petit landau (figure 4.3). Elles sont étonnement légères et solides.

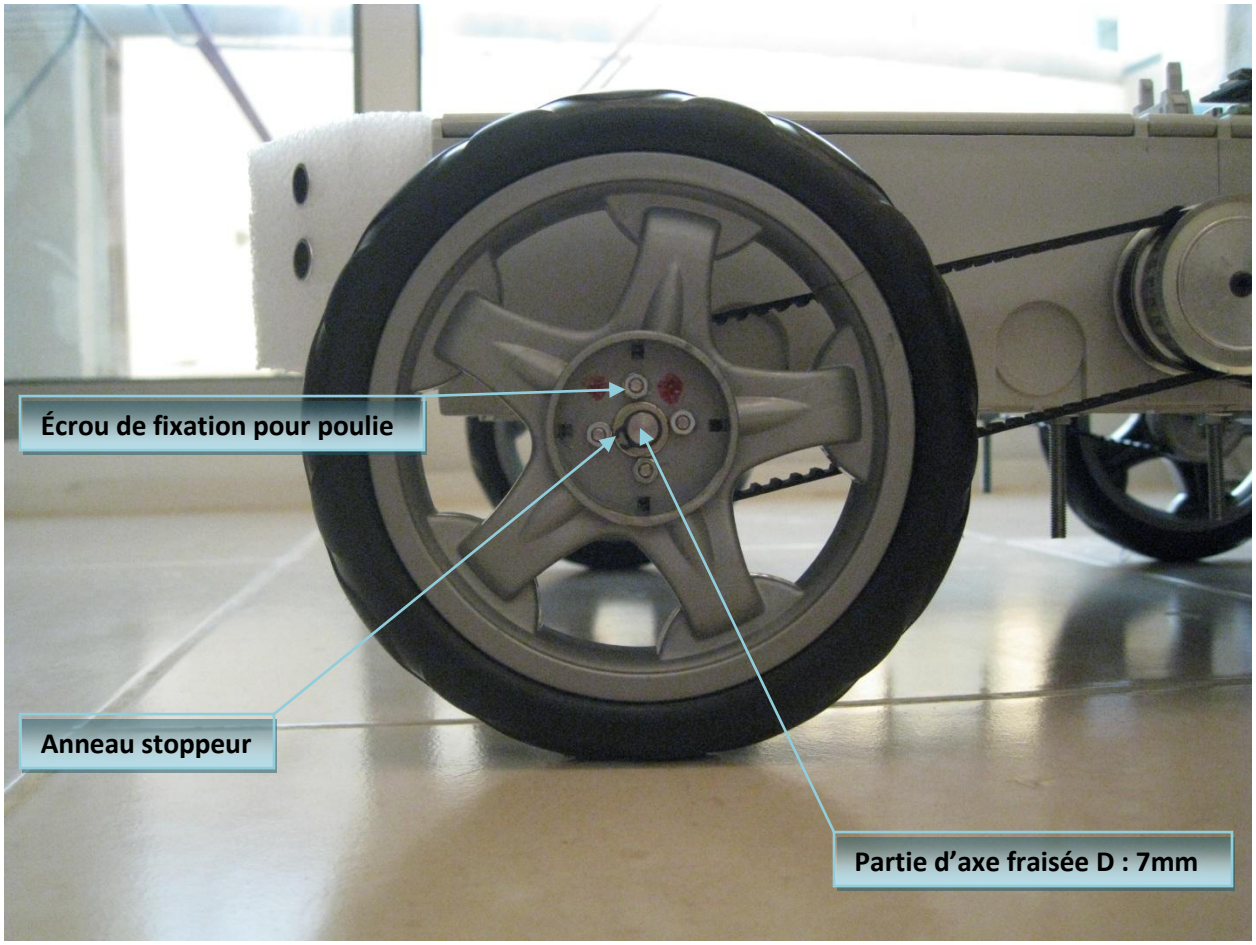



**Figure 4.3**, Vue extérieure de la roue.

À chacune de ces roues, nous avons attaché, du côté intérieur, une poulie dentée (figure 4.4) par laquelle le moteur entraîne la roue.

Ces poulies, en aluminium, sont fixées sur les roues par un jeu de 4 vis à écrous pour chacune (figures 4.3 et 4.4).

L'ensemble "roue + poulie" doit être fixé sur un axe. En effet, les roues étaient fabriquées pour être fixées sur un axe de diamètre 7mm. Or, nous disposons d'axes en "stainless steel" de 8 mm de diamètre, ce qui nous a amené à les adapter à nos roues en réduisant leurs diamètres, sur les 2 côtés, à 7 mm.

En plus, nous avons fraisé sur l'axe un petit canal par lequel se glisse un anneau stoppeur pour mieux tenir la roue à sa place (figure 4.3).

Enfin, les 2 axes, qui portent chacun deux roues avec leurs poulies, sont fixés au bas du châssis du robot (figure 4.4 et figure 4.5) par des colliers métalliques qui les maintiennent ensemble.

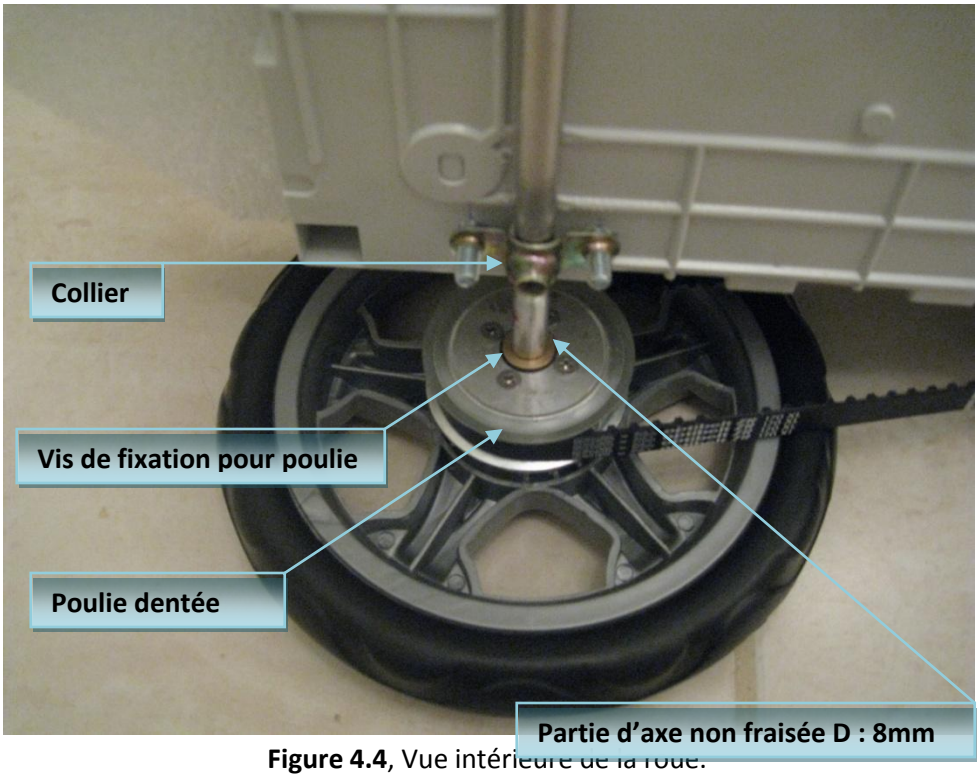



**Figure 4.4**, Vue intérieure de la roue.

**Figure 4.5**, Vue d'en-bas du chassis.

## *4.4 Les moteurs DC:*

Comme le montre la figure 4.5, chaque 2 roues d'un même coté sont entraînés par un moteur dédié; deux moteurs DC suffisent donc pour assurer la mobilité de l'appareil.

La figure 4.2 montre la fixation des deux moteurs, un collier est utilisé pour acheminer le câblage vers chaque moteur. Deux poulies dentées sont fixées sur l'axe du chaque moteur, reliée chacune par une ceinture dentée sur l'un des 2 roues (voir figure 4.1). La tension des deux ceintures sera réglée par un jeu de 2 vis à écrous qui servent aussi à la fixation des moteurs.

Les caractéristiques des moteurs à utiliser sont déterminées en se basant sur les spécifications de mouvement citées au paragraphe 4.3. Les détails du calcul sont résumés comme suit:

Le moment de la force est donné par: $Moment_{\,N.m} = Force_{\,N} \times Distance_{\,m}$ F 1
La distance, étant dans notre cas, le rayon de la roue, R = 7,5cm.

La force, étant celle de l'entraînement, est donnée par:
$Force_{\,N} = Masse_{\,Kg} \times Accélération_{\,m/sec^2}$ F 2

Étant donné que le poids de l'appareil est réparti sur les deux moteurs, chaque moteur doit être capable de faire déplacer une masse de 5 kg, ce qui donne une force maximale de 2.5 N pour une accélération maximale de 0,5 m/s$^2$ ($2{,}5_{\,N} = 5_{\,Kg} \times 0{,}5_{\,m/sec^2}$) qui correspond à un moment égal à $0{,}1875_{\,N.m} = 2{,}5_{\,N} \times 0{,}075_{\,m}$ .

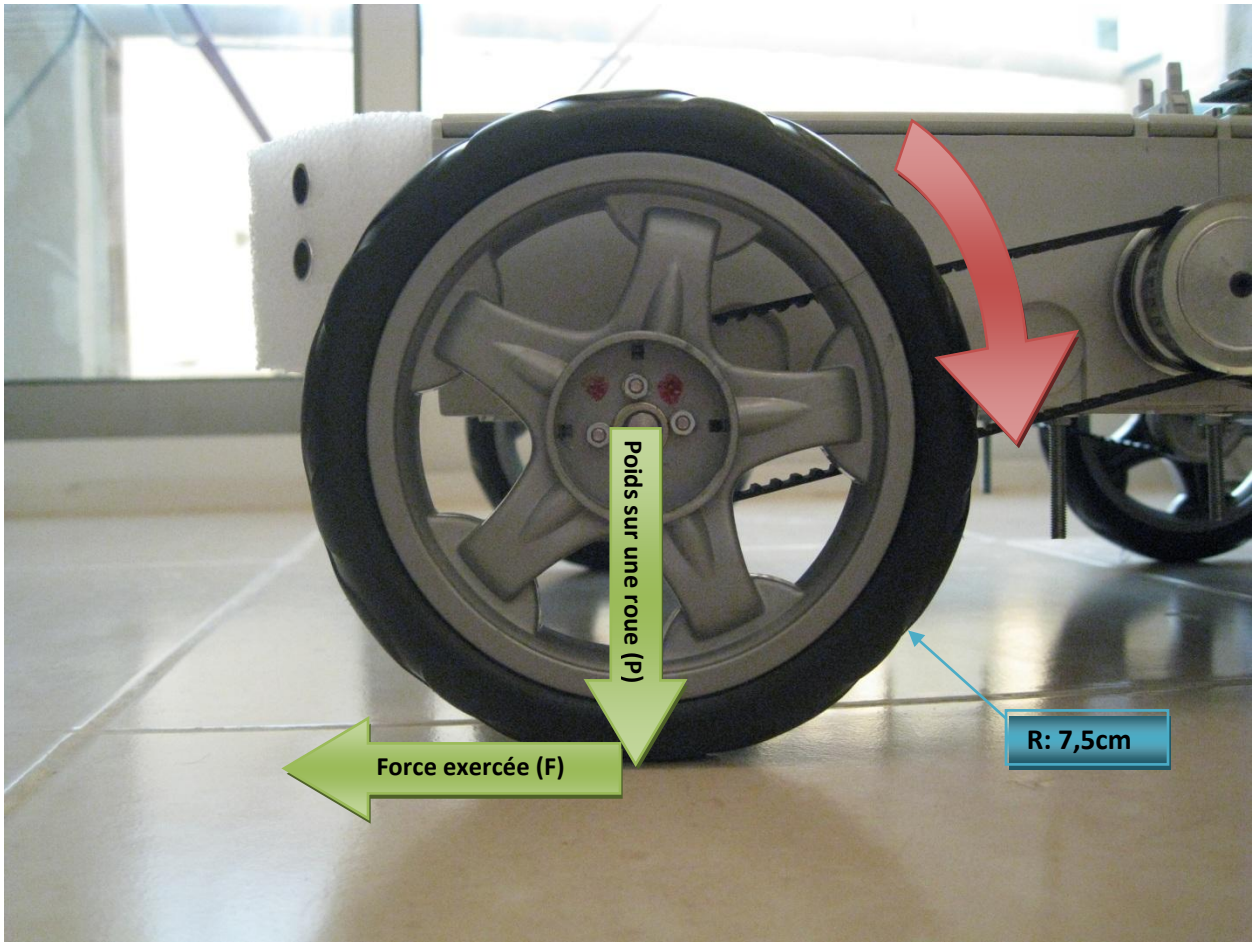



**Figure 4.6**, Représentation des forces exercées sur une roue.

Mais le fait que le robot pivote sur 4 roues la force exigée sera multipliée par un facteur qui dépend de

la situation, le type des roues et le type du chemin. Pour se donner un marge de manœuvre, nous avons choisi un facteur égal à 10.

Ainsi, nous nous basons sur un moment d'entraînement égal à 1,875 N.m

La vitesse de rotation maximale désirée pour la roue est calculée, en fonction de la vitesse linéaire maximale, par $\frac{0{,}5\ m/sec}{(0{,}15\times\pi)_m} = 1{,}061\ tours/sec$

Sur le marché libanais, nous avons trouvé un moteur DC qui peut délivrer un moment égal à 1,015 N.m avec une vitesse de rotation en sortie égale à 2,87 tours/sec. Ses autres caractéristiques étaient:

Référence : 3540K 024C
Tension d'alimentation : 24V.
Vitesse de rotation « RPM » : 5000 tours par minute.
Boite de vitesse Référence : 32/1-29:1.
Boîte de vitesse taux : 29/1.

Avec un système de poulie dentée réduisant la vitesse à 50%, la vitesse de rotation devienne 1,435 tour/sec ce qui correspond à une vitesse de 0,67 m/sec et un moment de 2,03 N.m, ce qui permet de l'adopter pour notre application.

**Figure 4.7**, les capteurs ultrasons fixés sur le pare-choc.

## *4.5 Le support des détecteurs:*

Pour protéger les côtés avant et arrière du châssis et pour avoir un support modelable sur lequel nous pouvons fixer les différents capteurs dont nous aurons besoin, nous avons choisi d'utiliser, pour servir

de pare-choc, deux morceaux de mousse spongieuse utilisés fréquemment dans l'emballage (figure 4.7).

## 4.6 Conclusion:

Dans ce chapitre, nous avons présenté l'approche que nous avons adoptée pour réaliser la partie mécanique d'un robot. Il est vrai que cette approche ne respecte pas les règles d'art dans ce domaine, mais nous pensons qu'elle est suffisante pour notre application et surtout elle a le mérite d'être simple.

# Chapitre 5 PARTIE ÉLECTRONIQUE

## *5.1. Introduction:*

Dans le Chapitre 1, nous avons défini des spécifications que notre produit doit avoir et que nous les rappelons ci-après:

- Être entraîné par 2 moteurs à courant continu, à travers lesquels son mouvement sera commandé.
- Être équipé par des dispositifs lui permettant d'éviter les obstacles (avoir des détecteurs ultrasonores, par exemple).
- Communiquer avec la manette de contrôle à travers une liaison hertzienne.
- Être évolutif dans le sens d'avoir la possibilité d'accueillir des interfaces additionnelles, pour y ajouter des nouvelles fonctionnalités (par exemple, D'autres capteurs ou détecteurs, un bras, alimentation photovoltaïque, … etc).
- Son alimentation électrique de base sera de 24V de laquelle on fait dériver les autres tenions nécessaires pour le fonctionnement du système.

Du point de vue électronique, le dispositif à concevoir et réaliser devra assurer, en plus des fonctionnalités assurées par l'unité centrale, les fonctions périphériques suivantes (figure 5.1) :

1. **L'alimentation :**

   La caractéristique principale d'une alimentation pour un système autonome à piles est l'efficacité et la fiabilité. Une seule alimentation de 24V suffit. Les autres tensions, de grandeurs plus faibles, en seront obtenues par découpage.

   Dans cette version du produit, en plus de 24V, on aura besoin d'une tension de 12V pour les détecteurs par ultrasons et de 5V pour la partie électronique.

   Cette alimentation fournie par des piles doit être rechargeable avec la possibilité de commuter entre ces dernières et la tension d'entrée nécessaires au chargement.

2. **L'entraînement des moteurs (Driver):**

   Les moteurs à courant continu DC sont souvent contrôlés par des signaux PWM. Toutefois, La fréquence du signal doit être déterminée en fonction du moteur utilisé. Pour une fréquence élevée, le moteur ne réagit pas à des faibles rapports cycliques tandis que, pour une fréquence basse, le mouvement du moteur sera discontinu.

   Pour cela, le circuit de commande des moteurs doit générer des signaux PWM dont la fréquence varie de 0 à 20 kHz, ce qui couvre presque toute la dynamique des fréquences utilisées pour contrôler les moteurs DC, sachant que la tension de sortie doit être égale à 24 V

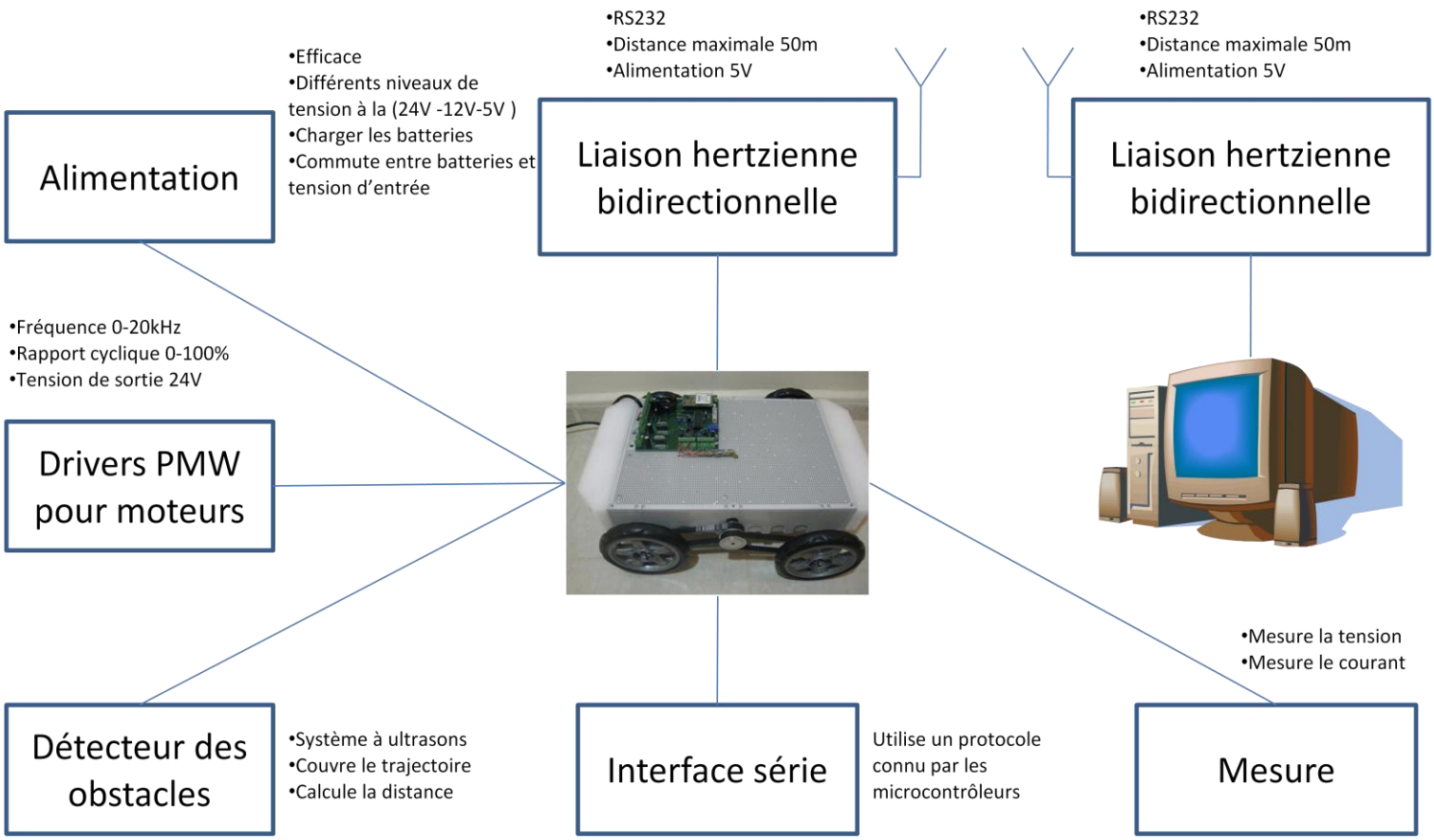



**Figure 5.1,** Diagramme représentative des fonctions

3. **La détection des obstacles:**

   Elle sera assurée par un détecteur ultrasonore fonctionnant à une fréquence de 40 kHz. Son rôle est de détecter les obstacles et de permettre de déterminer la distance qui les sépare du robot.

4. **La présence d'une interface série additionnelle:**

   Cette interface sera dédiée à l'utilisation future pour des éventuelles améliorations du robot. Elle permettrait d'ajouter des interfaces additionnelles. La seule contrainte est d'utiliser un protocole de communication reconnu par le système.

5. **Le contrôle de la consommation:**

   Dans un système autonome contrôlé via une liaison hertzienne, il est utile de savoir la consommation et la tension instantanée des batteries pour présager l'autonomie du système, par conséquence adapter le fonctionnement matériel et logiciel pour réduire la consommation en fonction du besoin.

6. **La gestion de la liaison hertzienne bidirectionnelle:**

   La liaison hertzienne doit être capable de transférer les données d'une manière bidirectionnelle

à travers un port RS232.

Le schéma 1 de l'annexe 2 est celui du contrôleur électronique du robot alors que le Schéma 2 montre le placement de ses différentes composantes sur le circuit imprimé tel qu'il sera réalisé. Tout au long de ce chapitre, nous nous référons à ces deux schémas.

### *5.2. Le microcontrôleur:*

Le cœur du contrôleur est constitué autour d'un microcontrôleur (U1). Notre choix s'est porté sur le PIC18F4431 ou PIC18F4331 de "Microchip". Les 2 chips sont identiques sauf en ce qui concerne la capacité de leur mémoire où sera stocké le code de l'application. Cette mémoire, est de type flash, est de 16 koctets pour le PIC18F4431 et de 8 koctets pour le PIC18F4331.

Nous avons choisi ces 2 références en se basant sur leurs caractéristiques suivantes:

- Le module PCPM (Power Control PWM Module) intégré : ce module génère des multiples signaux de type PWM qui sont utilisés pour contrôler les moteurs.
- Le module MFM (Motion Feedback Module) intégré: ce module sera dédié aux capteurs de mouvement.
- Le module EUSART (Enhanced Universal Synchronous Asynchronous Receive Transmitter) intégré qui peut assurer l'interface de la communication série (RS 232).
- Le port série synchrone (SSP -SYNCHRONOUS SERIAL PORT) : cette interface permet au microcontrôleur de communiquer avec d'autres processeurs par le protocole SPI ou par le protocole $I^2C$.
- Le module de conversion Analogique – Numérique à haut débit (10-Bit High-Speed Analog-to-Digital Converter (A/D) Module)qui contient 9 convertisseurs A/D.
- Les 36 entrées/sorties.

L'horloge interne du microcontrôleur est générée à l'aide d'un résonateur Crystal (Y1) ayant une fréquence de 20 MHz et de 2 condensateurs (C14) et (C15).

Le bouton poussoir (S1) permet de réinitialiser le microcontrôleur (Reset), tandis que la résistance (R1) et le condensateur (C13) le maintient à cet état pour un certain délai une fois (S1) est relâché.

Le connecteur J4 peut servir pour connecter des interfaces additionnelles au système.

RA2, RA3 et RA4 sont utilisés soit pour le MFM (Motion Feedback Module) ou soit comme des I/O.

RD1, RD2 et RD3 sont utilisés pour le SSP (SYNCHRONOUS SERIAL PORT); ils sont dédiés soit pour communiquer avec d'autres microcontrôleurs, soit pour être utilisés comme des I/O.

RC1 peut être utilisé pour le module CCP2 "Capture/Compare/PWM" ou comme un I/O. RC3 peut être utilisé comme entrée d'horloge pour "timer 5" (T5CKI) ou comme une entrée d'interruption externe (INT0) ou une I/O. RC4 peut être une entrée d'interruption externe (INT1) ou un I/O.

## 5.3. Contrôleur des moteurs:

Le contrôleur est une combinaison du module PCPM (Power Control PWM Module) intégré dans le microcontrôleur et les "H-Bridge" de référence LMD18200 (U2) et (U3) dédiés aux moteurs droite et gauche respectivement. Les connections de U2 et U3 au microcontrôleur sont identiques. Dans ce qui suit nous parlerons principalement de "U2".

Le module PCPM génère des signaux de type PWM ayant une fréquence de 12,5 kHz; ces signaux sont envoyés à U2 par les sorties RB1 et RB0. Les signaux de RB1 et RB0 sont complémentaires (voir La figure 5.3 où RB1 est en jaune et RB0 est en vert).

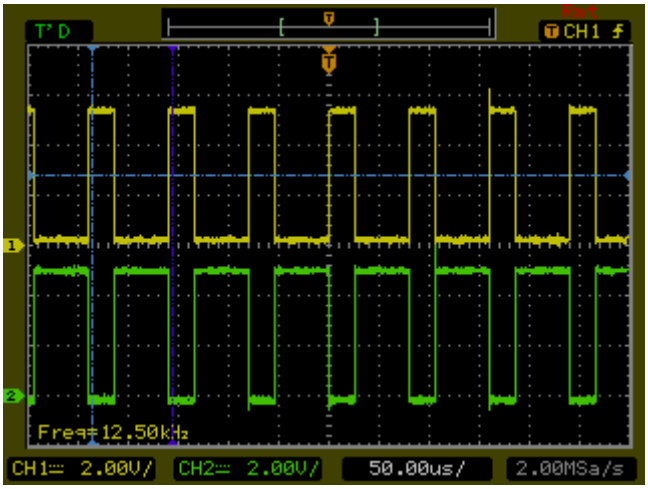



**Figure 5.3,** les signaux PMW du moteur droite.

Le sens de rotation du moteur est contrôlé par la sortie RD0. La logique de commande du LM18200 est présentée au tableau 5.1.

**Tableau 5.1, logique de commande du LM18200.**

| PWM | Dir. | Brake | Active Output Drivers | Etat |
|---|---|---|---|---|
| H | H | L | Source 1, Sink 2 | Moteur en avance |
| H | L | L | Sink 1, Source 2 | Moteur en marche arrière |
| L | X | L | Source 1, Source 2 | Moteur en arrêt (freinage) |
| H | H | H | Source 1, Source 2 | Moteur en arrêt (freinage) |
| H | L | H | Sink 1, Sink 2 | Moteur en arrêt (freinage) |
| L | X | H | NONE | Moteur débranché |

U2 et U3 sont liés, chacun, à la sortie d'un transistor à collecteur ouvert (TFO) qui devient passant une fois la température interne atteint 145°C. Ces 2 sorties connectées au 5V à travers la résistance (R5) forme une porte logique de type OU. Si l'un d'eux devient passant, l'entrée RD4 du microcontrôleur qui est configuré comme /FLTA passe à l'état bas. Par conséquence, les sorties PWM sont désactivées et une interruption interne est générée.

La sortie CSO d'U2 est une source de courant proportionnelle à la consommation du moteur. Cette sortie livre un courant égal à :$377_{\frac{\mu A}{A}} \times courant\ moteur$.

Une résistance (R3) de valeur 2,7 kΩ connectée au CSO génère une tension de :

$$377_{\frac{\mu A}{A}} \times 2{,}7_{K\Omega} \times courant\ moteur = 1{,}0179_{\frac{V}{A}} \times courant\ moteur.$$

Le condensateur C18 réduit le changement rapide de la tension causée par la discontinuité du signal PWM. Enfin, cette tension est appliquée à l'entré RA0 à travers un étage tampon créé par l'un des 4 amplificateurs opérationnels du LM6134 (U4-A).

La figure 5.4 montre les signaux PWM (RB1) en jaune et la sortie d'U4-A en vert qui présente la valeur du courant d'un moteur qui tourne à vide alors que la figure 5.5 présente celle d'un moteur intentionnellement bloqué.

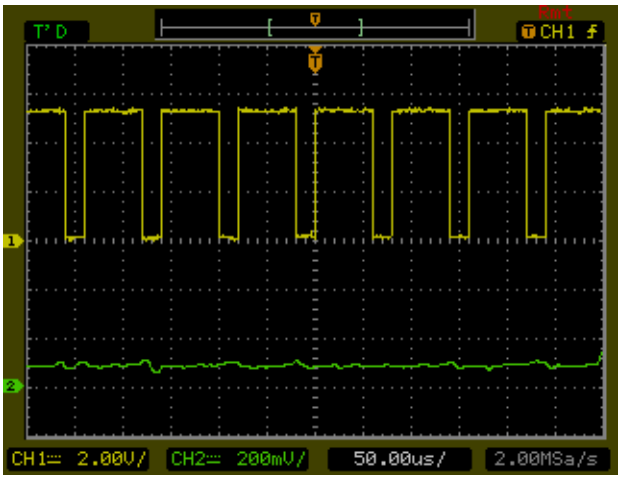



**Figure 5.4** En vert, le courant dissipé d'un moteur tournant à vide.

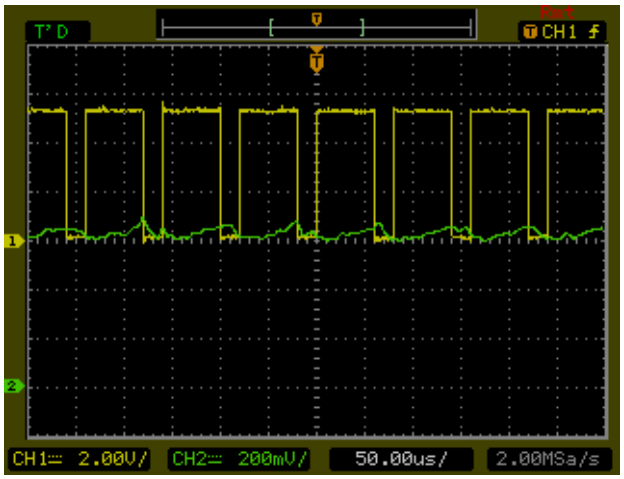



**Figure 5.5** En vert, le courant dissipé d'un moteur bloqué.

RA0 et RA1 sont configurés d'une manière permettant d'envoyer les signaux reçus vers le convertisseur A/D. Les valeurs sont ensuite envoyées par l'interface série vers l'ordinateur.

### *5.4. Mesurer la tension et le courant du contrôleur:*

Ce circuit a pour rôle de mesurer la tension des batteries et le courant maximal consommé par le système. Comme nous l'avons mentionné au paragraphe 5.1, ces valeurs nous permettent de bien contrôler la consommation.

Le circuit est alimenté à travers la jonction J3. Un pont diviseur réalisé par les résistances R10 et R11 réduit la tension de 24V à 1,2V. Cette tension est appliquée à un amplificateur opérationnel (U4-D), qui avec les résistances R13 et R12 constituent un étage amplificateur de gain égal à 2,5. Alors, pour une tension d'alimentation de 24V, la sortie de l'amplificateur sera égale à 3V.

Le condensateur C22 sert à éliminer les bruits et les ondulations. Sa présence avec l'amplificateur constitue un filtre passe-bas actif ayant une fréquence de coupure égale à $f_c = \frac{1}{2\pi \times R13 \times C22} = 2{,}25Hz$.

La sortie de l'amplificateur est ensuite envoyée au convertisseur A/D à travers l'entrée RE1 du

microcontrôleur.

Le courant total du système traverse la résistance R6. La chute de tension aux bornes de R6 est égale à $\Delta V = -0{,}01 \times It$. Pour un courant de 10A la chute de tension sera égale à -0,1V.

Cette tension est appliquée à un étage amplificateur inverseur réalisé autour d'U4-C, la tension à la sortie d'U4-C est égale à $V = -\frac{R9}{R8} \times \Delta V = \frac{R9}{R8} \times R6 \times It$. Ainsi, pour un courant est 10A la tension de sortie est égale à 1,9V qui est appliquée au convertisseur A/D à travers RE0 du microcontrôleur.

### 5.5. La liaison hertzienne:

La liaison hertzienne est réalisée par le module émetteur/récepteur 9XCite (T1) qui permet une communication "point-à-point" avec un autre module du même type.

RC6 et RC7 de microcontrôleur sont les TX et RX de l'interface (EUSART) respectivement. Les informations reçues par T1 (DO) sont envoyées au RC7 et celles venant de RC6 sont envoyées au T1 (DI).

Le contrôle de la communication sera fait à travers l'entrée RB7 connectée à /CTS et la sotie RC5 connectée à /RTS. On peut mettre T1 à l'état "Sleep" par la sortie RE2 pour économiser l'énergie. Le bouton poussoir (S2) initialise les paramètres de T1 à la configuration initiale du constructeur.

La LED (D2) s'allume à la présence de l'alimentation et s'éteint si T1 est à l'état de transmission. La LED (D3) s'allume si T2 est à l'état de réception.

### 5.6. Les détecteurs à ultrasons:

Le système de détection à ultrasons est constitué de 8 paires de détecteurs contenant, chacune, un émetteur et un récepteur à ultrasons. La sélection d'une paire se fait à travers le multiplexeur des émetteurs CD4051B (U6) et par le multiplexeur des récepteurs ADG508A (U5). U5 et U6 possèdent 3 entrées binaires qui sont utilisées pour la sélection de l'un des 8 canaux. Les 2 multiplexeurs sont alimentés par une tension de 12V.

Les sorties RB4, RB5 et RB6 sont dédiées à la sélection d'une paire émetteur/récepteur. Ces sorties sont converties de 5V à 12V par les trois transistors U7-A, U7-B et U7-C intégrés dans U7 associés aux résistances de couplage R27, R28 et R29, avant d'être appliquées aux entrées binaires U5 et U6. Pour chaque émetteur sélectionné par U6, un récepteur convenable est sélectionné par U5.

La sortie RD6 sert à désactiver U6 pour réduire la consommation, cette sortie est convertie de 5V à 12V par le transistor U7-D et la résistance R30. De même, la sortie RD5 sert à désactiver U5.

Le MC34151 (U9) est principalement dédié à contrôler les transistors à commutation rapide de type MOSFET ou généralement les charges capacitives. Notons que l'état haut dans ce circuit correspond à une tension supérieure ou égale à 1,75V et ce, quelle que soit la tension d'alimentation.

Les signaux ultrasonores de 40 kHz sont générés par le module CCP interne du microcontrôleur et sont envoyés à U9-A à travers la sortie RC2. Les signaux à la sortie d'U9-A sont inversés par U9-B avant qu'ils

soient appliqués à l'émetteur convenable par U6, sachant que le signal ultrasonore est envoyé durant 0,5 ms.

La figure 5.6 montre que, durant 0,5 ms, 20 impulsions dont la fréquence égale à 40 kHz sont émises suivies d'une période de silence égale à 1 ms pendant laquelle la détection est désactivée.

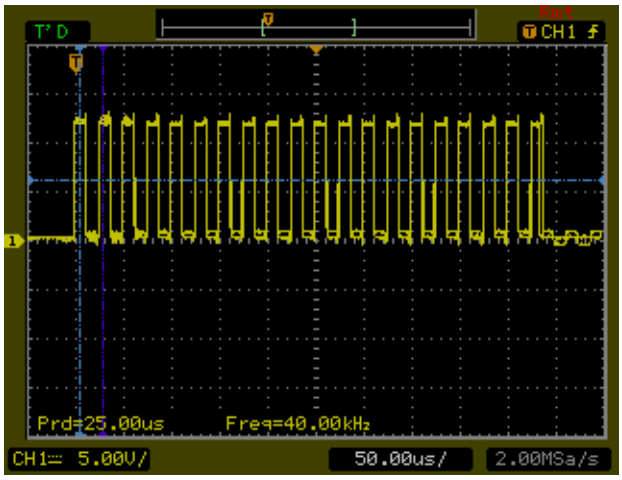



**Figure 5.6** Impulsions correspondantes au signal ultrasonore de 40 kHz.

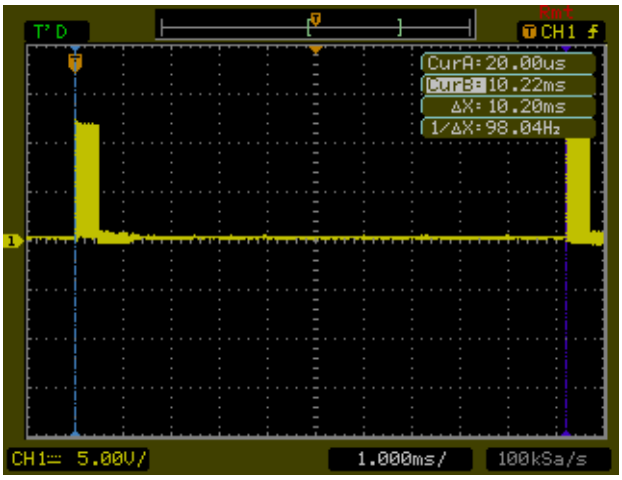



**Figure 5.7** Signal ultrasonore émis: période entre deux émissions consécutives.

Cette période de silence est justifiée par le fait que les émetteurs à ultrasons utilisés tendent à vibrer après la coupure du signal (La figure 5.7 montre le bruit dû à cette vibration pendant la période de silence de 1ms); or, ces vibrations sont détectables par les récepteurs, ce qui risque de faire générer une détection fausse. D'où la nécessité de cette période de silence.

Notons que cette période augmente la distance minimale détectable par le système qui est donnée par $\frac{(0{,}0005+0{,}001)}{2}_{sec} \times 328\frac{m}{sec} = 0{,}246m$ au lieu qu'elle soit de 0,082 m si on n'introduit pas cette période de silence.

Le système commence à saisir les données après 1 ms de la dernière impulsion envoyée. Cet état dure 8,7 ms ce qui correspond à une distance maximale donnée par $\frac{(0{,}001+0{,}0087)}{2}_{sec} \times 328\frac{m}{sec} = 1{,}59m$.

Comme le montre la figure 5.7, le basculement d'un détecteur à un autre se fait en 10,2 ms. Pour faire le tour des 8 détecteurs, nous aurons besoin de 81,6 ms, ce qui correspond à une fréquence d'émission-réception de 12 fois par seconde pour chaque 12 détecteur.

Les signaux captés par les récepteurs sont envoyés par U5 au premier étage amplificateur réalisé par le transistor Q1 qui est monté en émetteur commun. Cet amplificateur peut fournir un gain égal à 135.

La figure 5.8 montre en jaune le signal envoyé par l'émetteur et le signal au collecteur de Q1 en vert pour un objet éloigné d'un mètre.

Ce signal est ensuite appliqué au circuit amplificateur opérationnel réalisé par U8-A, C25 élimine la tension continue venant du collecteur et laisse passer seulement les ondulations.

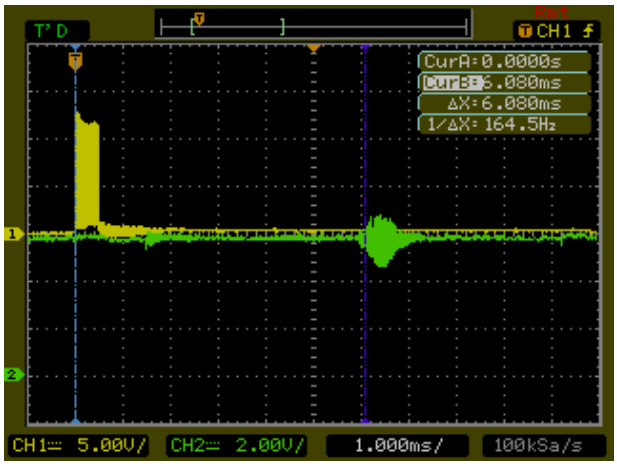



**Figure 5.8** En vert le signal ultrasonore détecté à la sortie de Q1.

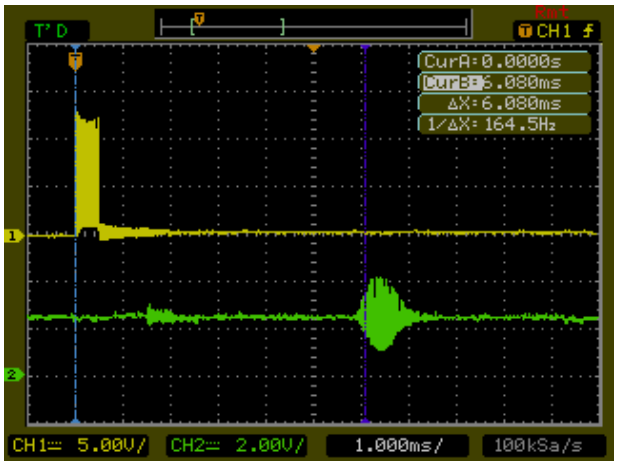



**Figure 5.9** En vert le signal ultrasonore détecté à la sortie U8-A.

Lors de l'absence d'un signal détecté, la tension à la sortie d'U8-A est fixée à 2,5V par le pont diviseur réalisé par R21 et R22. Les ondulations détectées venant de Q1 sont ensuite amplifiées par un gain égal à $G = \frac{R19+R20}{R18}$. La résistance variable R20 nous permet de modifier le gain de 1 jusqu'à 21 selon le besoin. Ce signal amplifié s'ajoute à la tension 2,5V à la sortie d'U8-A. La figure 5.9 compare le signal envoyé en jaune et celui détecté en vert à la sortie d'U8-A

U8-B et les résistances R23, R24 et R25 constituent un comparateur dont la tension de seuil est égale à $5 \times \frac{R25}{R23+R24}$. La tension de seuil du comparateur est contrôlée par R24, ceci qui assure une bonne précision du comparateur.

En absence d'un signal détecté, la sortie du comparateur est à l'état haut. En présence d'un obstacle , la tension des ondulations (en jaune, dans la figure 5.10)) devient plus petite que celle du seuil (en vert) la sortie d'U8-B bascule à l'état bas. La Figure 5.11 montre le signal à l'entrée du comparateur en jaune et celui à la sortie d'U8-B en vert.

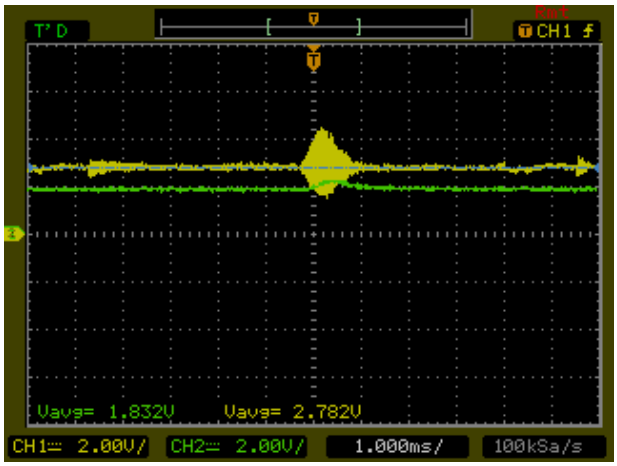



**Figure 5.10** Signal détecté en présence d'un obstacle (en jaune) comparé à la tension du seuil (en vert).

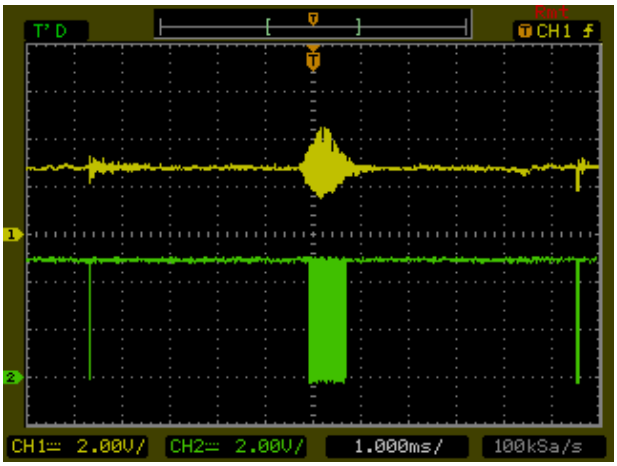



**Figure 5.11** Sortie du comparateur (en vert) en cas de détection.

La figure 5.12 présente le signal ultrasonore appliqué à l'émetteur (en jaune) et le signal réfléchi par un objet éloigné d'un mètre mesuré à la sortie d'U8-B (en vert). Cette figure montre un écart de 6,08ms entre le début du signal reçu et la première impulsion envoyée. À partir de cette mesure, nous pouvons calculer la distance de l'objet par $D = \frac{0{,}00608_{sec}}{2} \times 328_{\frac{m}{sec}} = 0{,}997m$ ce qui correspond à une précision de mesure à $10^{-3}$ près

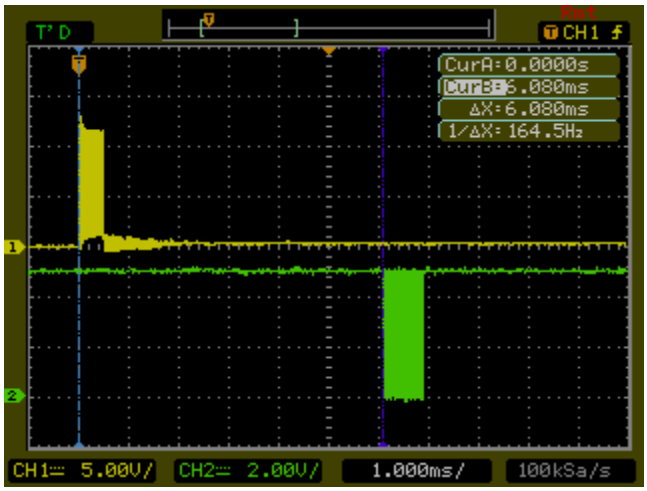



**Figure 5.12,** Le signal PMW envoyé (en jaune) et celui reçu (en vert) par le capteur correspondant à un objet éloigné d'un mètre du capteur.

Le signal fourni à la sortie U8-B est enfin appliqué à RC0 du microcontrôleur qui correspond à l'entrée d'un compteur. Ce compteur s'incrémente à chaque passage du signal d'entrée à l'état bas. Quand le nombre des impulsions détectées atteint 10, une interruption interne est générée pour enregistrer la distance parcourue. Le rôle du compteur est important pour le système car il sert à éliminer les fausses détections qui peuvent être causées par le bruit.

## *5.7. Les alimentations:*

La tension 5V est obtenue par le régulateur à découpage PT6302 (U11), tandis que la tension 12V est obtenue par le PT6304 (U10). En cas du besoin, le MOSFET 2N7000 (Q2) sert à désactiver U10 pour réduire la consommation. (U10) et (U11) peuvent fournir un courant de sortie égal à 3A chacun. Les 2 tensions de 5 et 12V sont aussi disponibles à la jonction J5 où on a utilisé la tension 12V afin d'alimenter la caméra.

Les deux batteries utilisées sont de type Plomb-acide, 12V de capacité 9Ah chacune. Elles sont montées en série pour fournir une alimentation égale à 24V. Elles peuvent être connectées directement à la jonction J3. Lorsque les batteries sont complètement déchargées, il faut les détacher afin de les charger. Ceci constitue une limitation surtout lorsqu'une opération continuelle est nécessaire.

La figure 5.13 présente la photo du circuit d'alimentation 24V, 10A utilisé. Ce circuit est disponible sur le marché. Il est alimenté par une tension de 24V AC donc, un transformateur externe 230V AC/24V AC est nécessaire. Ce circuit charge les batteries tant que l'entrée est alimentée et il fournit une sortie de 24V DC utilisée par le robot. En l'absence de la tension d'entrée, la sortie du circuit est fournie par les batteries.

**Figure 5.13,** La carte d'alimentation.

## *5.8. L'interface PC:*

Cette interface (photo de la figure 5.14), dont le rôle est d'assurer la communication entre l'ordinateur à travers son port série RS232 d'un côté et le robot à travers la liaison hertzienne, est décrite par le schéma électrique no. 3 de l'annexe 2, alors que le schéma 4 du même annexe montre l'emplacement de ses composantes sur le circuit imprimé.

La connexion à l'ordinateur se fait par le connecteur DB9 (J1). Le MAX232 (U1) traduit les bits reçus en 0-5V et ensuite les envoie au robot à travers le 9XCITE (T1).

Le circuit est alimenté par une tension de 9 à 12V DC à travers J2, cette tension est réglée à 5V par le régulateur 7805 (Z1).

Le MAX232 est un circuit intégré, dont le rôle est de convertir les signaux envoyés d'un port série RS-232 en signaux appropriés pour une utilisation dans les circuits logiques numériques et compatibles avec la technologie TTL. Le MAX232 convertit généralement les signaux RX, TX, CTS et RTS.

Le tableau 5.2 représente les niveaux des signaux d'un port série RS-232 et les signaux TTL correspondants.

**Tableau 5.2, les niveaux des signaux d'un port série RS-232 et les signaux TTL correspondants.**

| Type du signal | Port série RS-232 | Technologie TTL |
|---|---|---|
| Signal de données (Rx/Tx) état 0 | +3V à +15V | 0V |
| Signal de données (Rx/Tx) état 1 | -3V à -15V | 5V |
| Signal de contrôle (RTS/CTS/DTR/DSR) état 0 | -3V à -15V | 5V |
| Signal de contrôle (RTS/CTS/DTR/DSR) état 1 | +3V à +15V | 0V |


**Figure 5.14,** montre l'interface PC.

## *5.9. Le programme en assembleur:*

La partie logicielle du système est celle qui assure le contrôle, la commande et la gestion de la communication de notre robot. On présente, dans ce qui suit, les principales fonctions du programme, à travers la description de leurs organigrammes respectifs. Le programme détaillé est fourni en annexe 3.

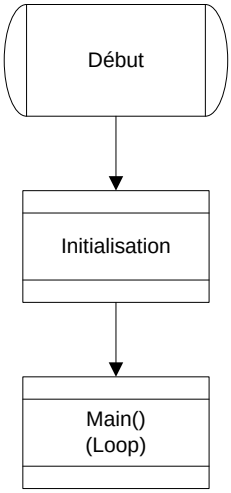



**Diagramme 5.1 :** Organigramme du programme principal

Notons, au début, que le programme tout entier est subdivisé en deux parties, comme le montre le diagramme 5.1 :

- La partie "Initialisation", qui sera exécutée une seule fois au début du lancement du programme,

et dans laquelle seront configurées et initialisées les interfaces et les modules du microcontrôleur avant de se mettre dans un état d'une boucle infinie.

- La partie "Boucle" (Loop), qui sera exécutée d'une façon continue (boucle infinie) tant que le programme s'exécute, et dans laquelle le microcontrôleur attend et gère les interruptions.

**a. La procédure d'identification de l'interruption (Diagramme 5.2):**

Cette procédure cherche l'origine de l'interruption et exécute la procédure convenable. Les interruptions prises en charge sont :

- "TMR0 Overflow", cette interruption est générée par la minuterie TMR0 pour indiquer que les signaux radar sont envoyés. Ceci provient de deux événements possibles:
  - La durée de silence est terminée ou,
  - le temps de l'état d'écoute est fini.
- "TMR1 Overflow", cette interruption est générée par la minuterie TMR1 afin d'indiquer que 10 impulsions radar sont reçues.
- Conversion A/D, indique que la conversion est finie.
- USART RX, indique qu'un octet est reçu par le port série.
- USART TX, indique qu'un octet est envoyé par le port série.
- Interruption inconnue : cette procédure s'exécute lorsque l'interruption est inconnue.

**b. Procédure de l'interruption TRM0_INT (Diagramme 5.3):**

Cette procédure teste si l'émission de signal radar est activée (PWM): Si oui, on la désactive et on met le radar en état de silence. Sinon, on teste alors dans quel état se trouve le signal radar; S'il s'agit de l'état silence, on change l'état du radar à l'état d'écoute puis on active la réception et TMR1. Si c'est l'état d'écoute, on désactive la réception et TMR1 et on émet un nouveau signal radar via l'émetteur suivant.

**c. Procédure de l'interruption TRM1_INT (Diagramme 5.4):**

L'interruption est générée par TMR1 quand 10 impulsions sont reçues par RC0, ce qui signifie la détection d'un obstacle. Cette procédure doit déterminer quelle paire de détecteurs radar a été activée, enregistrer la valeur de minuterie TRM0 dans la mémoire convenable pour être envoyée ensuite via le port série.

**d. Procédure de l'interruption de conversion A/D (Diagramme 5.5):**

Quand le convertisseur A/D convertit les 4 valeurs analogiques relatives à la lecture des "courants" des moteurs gauche et droit, du courant et de la tension de la batterie, il génère une interruption pour enregistrer les valeurs dans les mémoires correspondantes.

**e. Procédure de l'interruption de lecture du port série (Diagramme 5.6):**

Le microcontrôleur reçoit 2 types de commandes de l'extérieur:

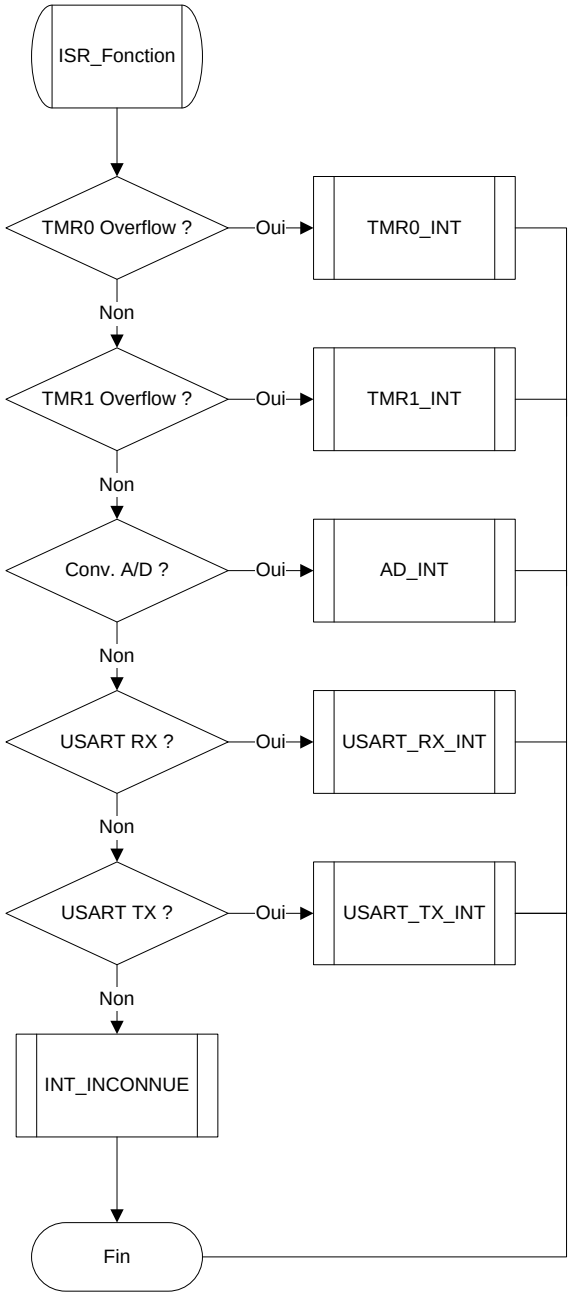



**Diagramme 5.2 :** Organigramme de la fonction de gestion des interruptions

- Un octet de valeur égale à 10 en hexadécimal qui signifie une commande d'arrêt du mouvement.
- Une commande de mouvement formée de 3 octets : Le premier égal à 11 en hexadécimal pour indiquer qu'il s'agit d'une commande de mouvement et pour mettre le microcontrôleur en état d'attente pour recevoir les 2 autres octets dédiés à contrôler la vitesse et la direction des deux moteurs. Les valeurs de la vitesse sont limitées entre -100 et 100. Le signe indique la direction de mouvement tandis que la vitesse varie de 0 et 100 qui correspond à un rapport cyclique entre 0 et 100% respectivement.

Si la commande reçue correspond à un arrêt, la procédure désactive toutes les interfaces sauf celui

du port série.

Si elle correspond à un mouvement la procédure active les interfaces et fixe les paramètres de mouvement des moteurs.

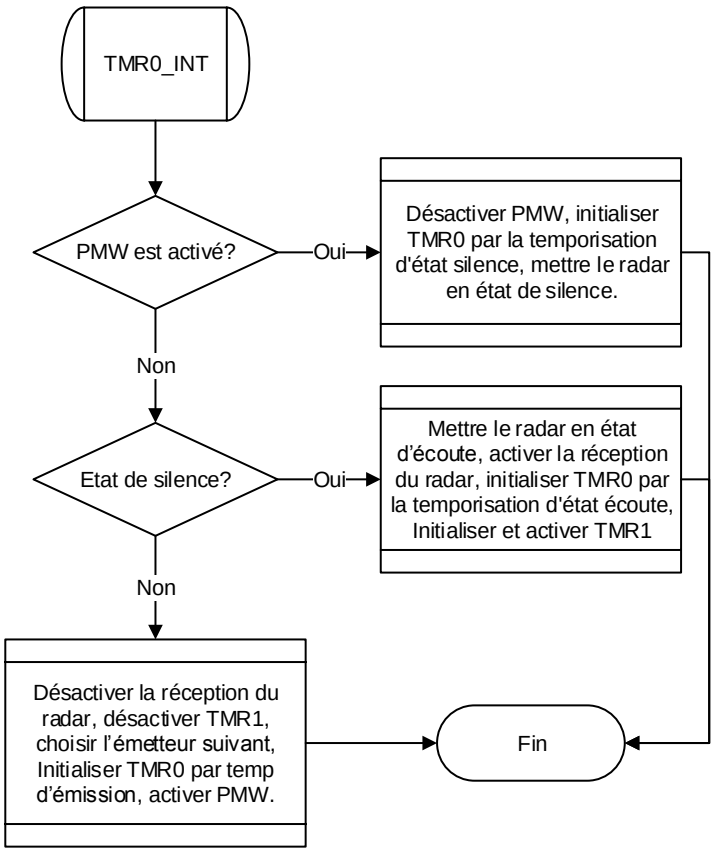



**Diagramme 5.3:** Organigramme de gestion de l'interruption TRM0_INT

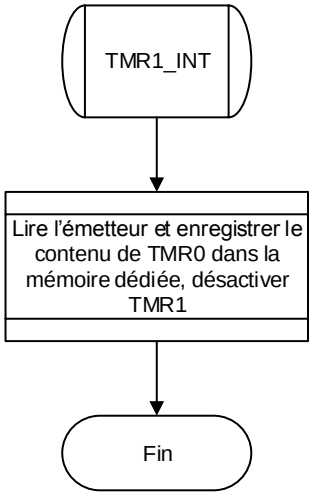



**Diagramme 5.4:** Organigramme de gestion de l'interruption TRM1_INT

Dans l'autre sens, le robot envoie 21 octets à travers le port série vers l'ordinateur. Ceux-ci correspondent à :

- État du robot (activé, désactivé, interruption inconnue) : 1 octet

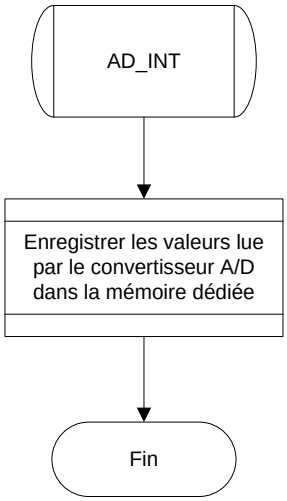



**Diagramme 5.5:** Organigramme de gestion de l'interruption du convertisseur

- 16 octets qui contiennent les valeurs fournies par le radar: en effet le microcontrôleur envoie pour chaque détecteur les 2 octets du registre du compteur TMR0 qui correspondent au temps d'écoute lus durant une interruption de TMR1 qui sont ensuite traités par le programme dans l'ordinateur pour obtenir la distance en mètres.
- 4 octets qui contiennent les valeurs numériques converties par le convertisseur A/D et qui correspondent aux valeurs électriques de la batterie et des moteurs.

Le tableau 5.3 décrit comment les informations sont représentées par les octets envoyés.

Après la réception du dernier octet de la commande du mouvement, cette procédure commence la transmission des octets en activant TX et en envoyant le premier octet de la séquence sachant que la transmission des autres octets sera complétée par la procédure suivante.

**f. Procédure d'interruption pour l'écriture sur le port série (Diagramme 5.7):**

Cette procédure continue l'envoi des 20 octets d'états. La transmission se désactive après l'envoi du dernier octet.

**g. Procédure d'une interruption inconnue (Diagramme 5.8):**

En cas de génération d'une interruption différente de celles prises en charge par le programme, cette procédure annule les drapeaux positionnés par l'interruption et initialise le premier octet à envoyer par le code d'interruption inconnue.

Le code d'interruption inconnue n'a aucune implication au fonctionnement du robot. Cette procédure est ajoutée pour offrir au robot un moyen de déterminer l'anomalie dans le système durant la période de la conception.

## *5.10. Conclusion:*

À travers ce chapitre, nous avons présenté la partie électronique que nous avons réalisée dans l'objectif de fournir à notre appareil une très grande flexibilité lui permettant d'accueillir, sans intervention

majeure, des interfaces additionnelles, des capteurs, … etc.

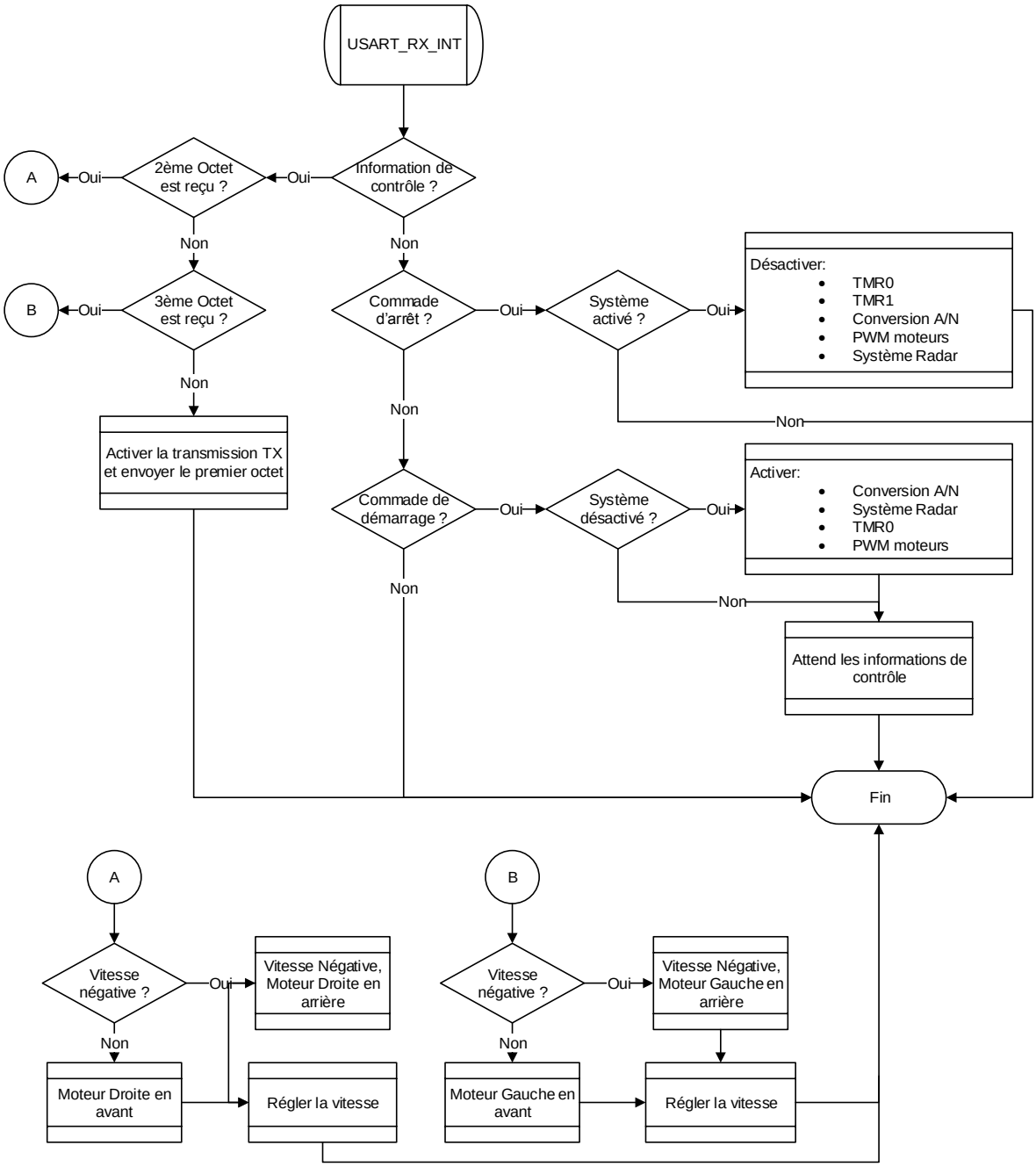



**Diagramme 5.6:** Organigramme de gestion de l'interruption de lecture du port série

Le programme du microcontrôleur que nous avons écrit peut être facilement modifiable pour optimiser différents aspects tels que l'efficacité, la fonctionnalité ou le comportement du robot.

Notons aussi, qu'on peut charger les batteries du robot, à n'importe quel instant, sans éteindre le robot, mais toutefois, au prix de la limitation de sa mobilité.

Notons enfin, que cette partie électronique peut être utilisée pour tout autre type de robot entraîné par deux moteurs.

**Tableau 5.3 Les informations transmises par le robot.**

| Octet | Registre dédiée | Nom de la valeur | Plage des valeurs en Hexadécimal | Description |
|---|---|---|---|---|
| 1 | _ | État | 10 ou 11 | 10 indique un fonctionnement normal, 11 indique une interruption inconnue |
| 2 | TMR0L | Distance du détecteur 1 | 55D2 à FFFF | 55D2 correspond à la distance minimale détectable (246 mm) et FFFF à la distance maximale détectable (1.591 m) |
| 3 | TMR0H | | | |
| 4 | TMR0L | Distance du détecteur 2 | | |
| 5 | TMR0H | | | |
| 6 | TMR0L | Distance du détecteur 3 | | |
| 7 | TMR0H | | | |
| 8 | TMR0L | Distance du détecteur 4 | | |
| 9 | TMR0H | | | |
| 10 | TMR0L | Distance du détecteur 5 | | |
| 11 | TMR0H | | | |
| 12 | TMR0L | Distance du détecteur 6 | | |
| 13 | TMR0H | | | |
| 14 | TMR0L | Distance du détecteur 7 | | |
| 15 | TMR0H | | | |
| 16 | TMR0L | Distance du détecteur 8 | | |
| 17 | TMR0H | | | |
| 18 | ADRESH | Courant du moteur droit | 0 à FF | 0 correspond au courant nul et FF au courant 4.91 A |
| 19 | ADRESH | Courant du moteur gauche | | |
| 20 | ADRESH | Courant absorbé par le robot | | 0 correspond au courant nul et FF au courant 26.31 A |
| 21 | ADRESH | Tension de la batterie | | 0 correspond à la tension nulle et FF à la tension 40 V |

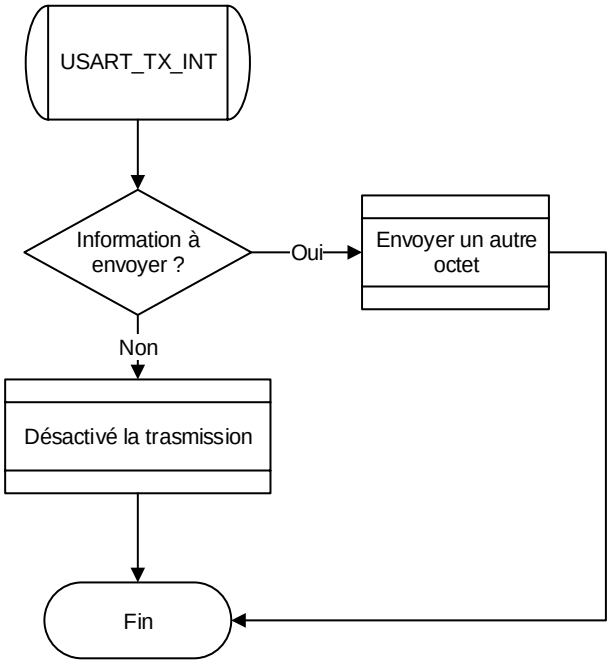



**Diagramme 5.7:** Organigramme de gestion de l'interruption l'écriture sur le port série

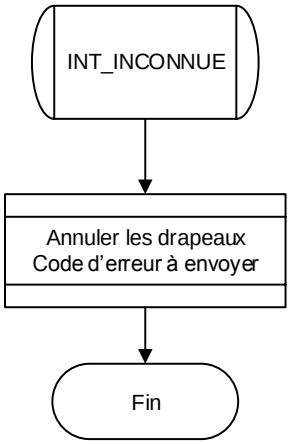



**Diagramme 5.8:** Organigramme de gestion d'une interruption inconnue

# Chapitre 6 : PARTIE LOGICIELLE

## 6.1 Introduction:

Pour donner à notre système une certaine autonomie pour se déplacer dans une trajectoire prédéfinie sans qu'il soit manipulé de l'extérieur, nous avons choisi d'équiper le robot par une caméra qui transmet des images réelles de l'environnement où il se déplace. Ces images seront ensuite traitées en temps réel par l'ordinateur de contrôle.

Comme outil d'analyse d'images, nous avons choisi de mettre en place et de tester la fiabilité d'un réseau de neurones entraîné par apprentissage. La construction, l'apprentissage et le test du réseau de neurones seront effectués, pour notre prototype, en utilisant la boîte à outils offerte par Matlab. En effet, Matlab offre une variété des outils qui nous aident à créer des réseaux personnalisés, ainsi que des fonctions d'apprentissage convenables à notre dispositif.

Pour simplifier l'usage de ces outils, nous allons concevoir des interfaces graphiques qui nous aideront à contrôler le robot soit manuellement soit via le réseau neuronal entraîné, et à manipuler les images utilisées dans la phase d'apprentissage.

Dans ce chapitre, nous décrivons le travail effectué sous Matlab aussi bien pour construire, entraîner et tester le réseau de neurones que pour développer et mettre en place l'interface graphique correspondante.

## 6.2 Le réseau neuronal artificiel:

Afin de concevoir le réseau neuronal, il faut déterminer au préalable ses dimensions, en particulier celles de l'entrée et de sortie et ceci en tenant compte de la nature des données à traiter. D'autre part les dimensions et l'architecture du réseau et le nombre des informations utilisées pour l'apprentissage affectent fortement le temps et les résultats d'entraînement.

Les images à traiter sont de 60x80 pixels contenant uniquement les informations de la luminance. Les valeurs des pixels de l'image sont les données à appliquer sur l'entrée du réseau sous forme d'un vecteur $x \in \Re^{4800\times1}$, donc une entrée de 4800 valeurs correspondant à la luminance de chaque pixel.

Le diagramme 6.1 montre l'architecture simplifiée du réseau utilisé.

La première couche statique du réseau utilisé (couche 1) sert à traiter les informations reçues pour en produire n1 valeurs caractéristiques à la sortie qui seront utilisées ensuite pour la classification. Cette valeur (n1) est choisie en tenant compte de la performance générale du réseau et la capacité de converger vers une solution dans un temps d'apprentissage raisonnable. En effet une grande valeur permet à cette couche de bien classifier les informations présentées à l'entrée mais le temps d'entraînement du réseau s'élève considérablement, surtout lorsque la quantité d'informations d'entrainement présentées à l'entrée est grande. D'autre part, pour une petite valeur de n1 le réseau ne

se converge pas vers une solution si le quantité des donnés d'apprentissage est élevée. n1 = 300 est choisi comme un compromis acceptable entre le temps d'apprentissage et la performance souhaitée.

La deuxième couche (couche 2), formée de 100 neurones, reçoit les sorties de la première et les classifie dans un nombre réduit de classes (100 au lieu de 300 prévu initialement). Cette réduction a principalement deux avantages, premièrement la réduction de la dimension d'entrée de la troisième couche ce qui réduirait le temps d'apprentissage sans réduire la performance du réseau, deuxièmement de permettre au réseau de mieux généraliser (ce qui le rend moins sensible au bruit et aux objets inconnus dans les images) durant la simulation.

Durant les premiers essais nous avons utilisés une seule couche à la place de la première et la deuxième qui est formée de 100 neurones, nous avons constatés que le réseau ne se converge pas vers une solution si le nombre des données d'apprentissage est grand (dans l'ordre de 150 images) et cela parce que le nombre total des poids synaptiques dans cette couche est relativement petit 4800 x 100 = 480.000 poids synaptiques. Tandis que dans la solution adoptée finalement nous avons 4800 x 300 = 1.440.000 poids synaptiques dans la première couche.

La dernière couche (couche 3) reçoit les informations de la couche 2 d'une manière directe et à travers 4 unités de retard. Ces 5 connections permettent au réseau de prendre des décisions basées sur l'image actuelle ainsi que sur les 4 dernières images reçues précédemment.

Cette possibilité de décider en tenant compte des informations déjà acquises précédemment offre au réseau une caractéristique dynamique.

Ainsi, les couches 2 et 3 sont interconnectées à travers 5 connections dont chacune contiennent 100x42 poids synaptiques, ce qui fait au total 5x100x42.

En l'absence de la couche intermédiaire (couche 2), le nombre des poids synaptiques sera 3 fois plus grand, ce qui augmenterait le temps d'apprentissage nécessaire, d'où l'intérêt de cette couche.

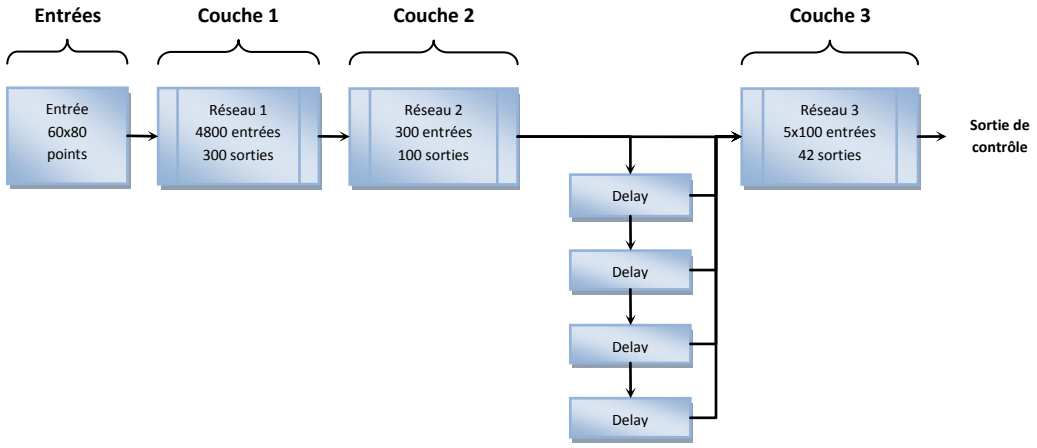



**Diagramme 6.1,** architecture du réseau neuronal utilisé dans l'application. Le tableau 6.1 liste ces 42 sorties possibles.

La couche 3 de sortie contient 42 neurones donc 42 sorties; elle constitue l'étage final du réseau dédié au contrôle des moteurs. Chaque sortie correspond à l'une des 42 combinaisons possibles que nous

avons fixées pour les trois paramètres suivants:

- Valeur du rapport cyclique (le pas est de 10%) du signal PWM qui sera appliqué à l'un des 2 moteurs
- le sens du mouvement (Avant/Arrière)
- Le moteur concerné (Droit/Gauche).

**Tableau 6.1, Relations entre les sorties du réseau et les commandes appliquées aux moteurs.**

| Sortie activée | PWM rapport cyclique | Direction | Moteur |
|---|---|---|---|
| 1 | 10% | Avant | Gauche |
| 2 | 20% | Avant | Gauche |
| 3 | 30% | Avant | Gauche |
| 4 | 40% | Avant | Gauche |
| 5 | 50% | Avant | Gauche |
| 6 | 60% | Avant | Gauche |
| 7 | 70% | Avant | Gauche |
| 8 | 80% | Avant | Gauche |
| 9 | 90% | Avant | Gauche |
| 10 | 100% | Avant | Gauche |
| 11 | 10% | Arrière | Gauche |
| 12 | 20% | Arrière | Gauche |
| 13 | 30% | Arrière | Gauche |
| 14 | 40% | Arrière | Gauche |
| 15 | 50% | Arrière | Gauche |
| 16 | 60% | Arrière | Gauche |
| 17 | 70% | Arrière | Gauche |
| 18 | 80% | Arrière | Gauche |
| 19 | 90% | Arrière | Gauche |
| 20 | 100% | Arrière | Gauche |
| 21 | 0% | - | Gauche |
| 22 | 10% | Avant | Droite |
| 23 | 20% | Avant | Droite |
| 24 | 30% | Avant | Droite |
| 25 | 40% | Avant | Droite |
| 26 | 50% | Avant | Droite |
| 27 | 60% | Avant | Droite |
| 28 | 70% | Avant | Droite |
| 29 | 80% | Avant | Droite |
| 30 | 90% | Avant | Droite |
| 31 | 100% | Avant | Droite |
| 32 | 10% | Arrière | Droite |
| 33 | 20% | Arrière | Droite |
| 34 | 30% | Arrière | Droite |
| 35 | 40% | Arrière | Droite |
| 36 | 50% | Arrière | Droite |
| 37 | 60% | Arrière | Droite |
| 38 | 70% | Arrière | Droite |
| 39 | 80% | Arrière | Droite |
| 40 | 90% | Arrière | Droite |
| 41 | 100% | Arrière | Droite |
| 42 | 0% | - | Droite |

Le fichier Matlab (m-file) "initnet" est utilisé pour créer et configurer le réseau décrit ci-dessus. Il est ensuite sauvegardé dans un fichier de données (type .mat). Ce réseau sera ensuite entraîné lors de la phase d'apprentissage avant qu'il soit utilisé dans la phase opérationnelle de contrôle.

## 6.3 Les interfaces graphiques Homme-machine:

L'interface principale (figure 6.1) contient 3 boutons:

- "Connection",
- "Control" et,
- "Train".

La partie droite de l'interface est réservée à l'affichage des messages émis par le logiciel concernant l'état de la connexion au port série, le numéro de ce port et la vitesse de communication en Bauds.

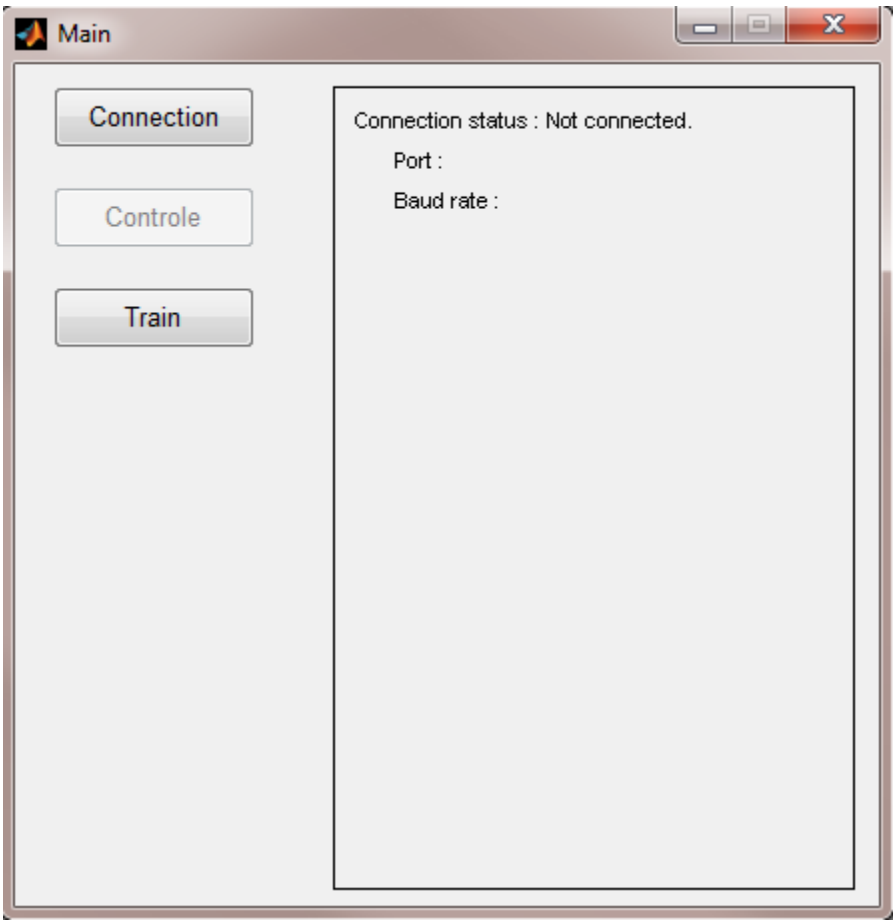



**Figure 6.1,** L'interface principale.

**Connexion :**

En appuyant sur le bouton "Connection", une autre interface graphique (GUI) s'ouvre pour permettre à l'utilisateur de choisir le port série convenable et de le configurer. La figure 6.2 montre cette GUI de connexion.

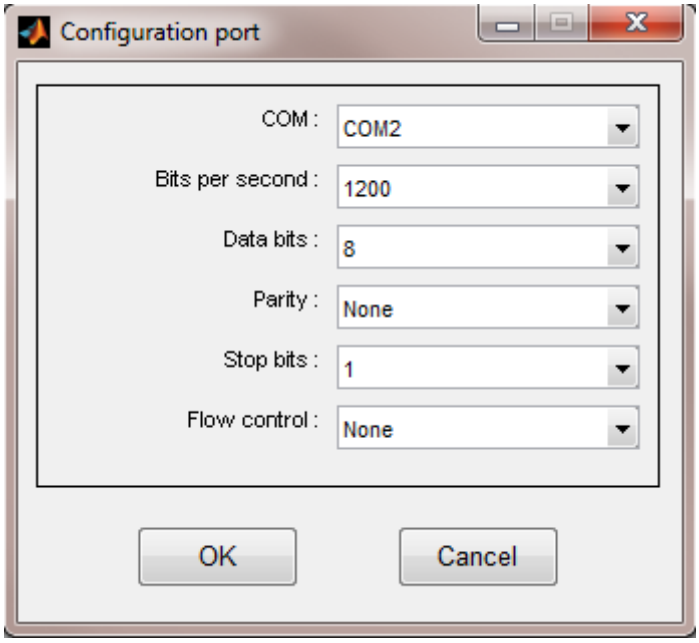



**Figure 6.2,** interface de connexion pour configurer le port série.

**Contrôle manuel :**

Après avoir configuré et validé (en appuyant sur "OK") le port série avec les paramètres convenables, l'interface de connexion se ferme et un message s'affiche sur la fenêtre de l'interface principale indiquant que le système est connecté (figure 6.3).

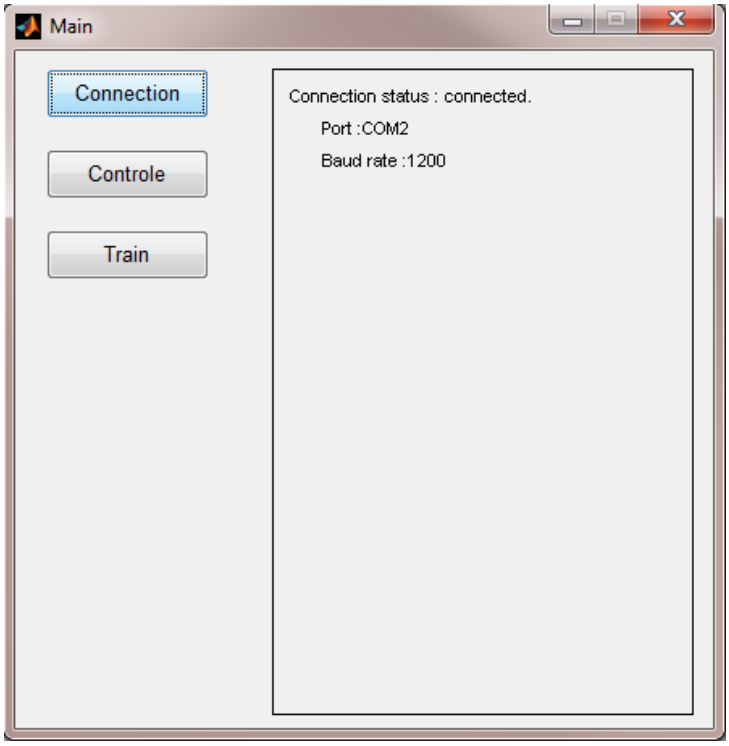



**Figure 6.3, l'**interface principale lorsque le port série est connecté.

Une fois le système connecté, le bouton "Control" s'active, ce qui permet à l'utilisateur de l'appuyer pour passer dans la phase de navigation du robot.

Dans ce cas, l'interface de contrôle s'ouvre (figure 6.4) avec une fenêtre d'affichage (figure 6.5) des images réelles provenant de la caméra du robot.

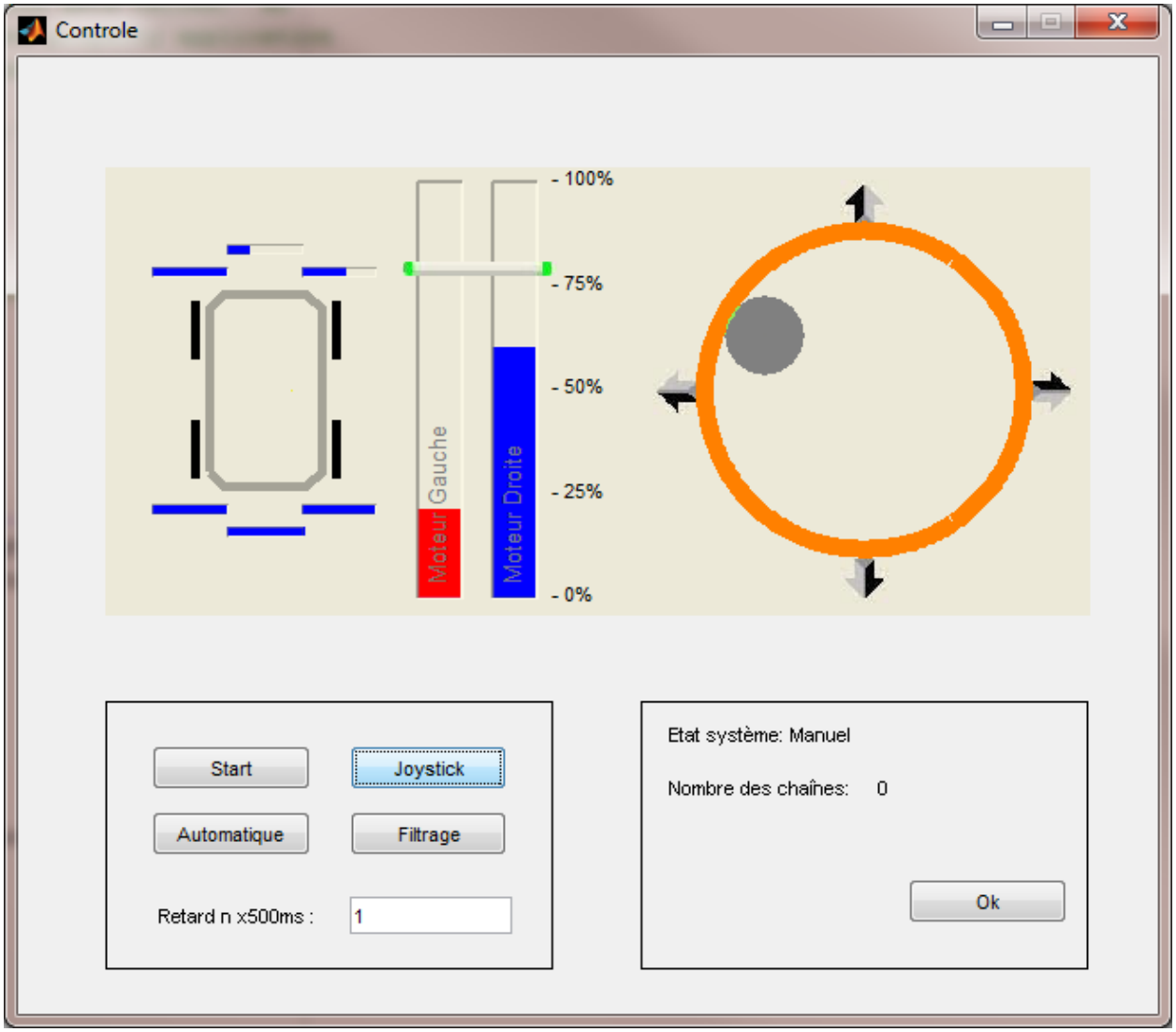



**Figure 6.4,** interface de contrôle.

Le contrôle manuel du robot se fait à l'aide d'une manette (Joystick) utilisée dans les jeux sur PC.

Ce contrôle manuel démarre lorsque le bouton "Joystick" est appuyé. Un message apparaît alors pour indiquer que le système fonctionne en mode manuel. Le cercle gris à l'intérieur du cercle orange indique la position courante du Joystick.

Les deux barres verticales colorées à côté du grand cercle visualisent les vitesses et le sens de rotation des 2 moteurs. La barre horizontale indique la limitation de vitesse que les moteurs ne peuvent pas dépasser.

La barre relative à un moteur est en bleu lorsque le moteur avance, elle est en rouge lorsqu'il recule.

Les 6 petites barres horizontales qui entourent le dessin schématique de l'engin permettent de

visualiser les distances qui séparent les 6 détecteurs ultrasonores des obstacles les plus proches. Lorsque la couleur de la barre est pleine, ceci veut dire qu'il n'y a pas d'obstacle détecté à une distance inférieur à 1,6 m. Sinon, la longueur colorée dans la barre se réduit proportionnellement à la distance qui sépare le détecteur de l'obstacle détecté.

**Entrainement du réseau de neurones :**

La navigation autonome de l'engin sur une trajectoire prédéfinie peut se faire via un pilotage automatique effectué par le réseau de neurones entraîné.

L'entraînement du réseau est réalisé en effectuant un pilotage manuel de l'engin tout au long de la trajectoire prédéfinie. Durant cette navigation, les images acquises à des intervalles de temps réguliers seront appliquées à l'entrée du réseau de neurones alors que les valeurs des vitesses courantes des 2 moteurs serviront comme informations sur les sorties souhaitées dans la situation courante. Ces paires d'entrée/sortie sont ainsi utilisées pour entrainer le réseau neuronal.

L'intervalle entre 2 images consécutives est fixé en introduisant une valeur entière n dans l'emplacement nommé "Retard nx500ms". Cet intervalle vaut alors, en secondes, 0.5 x n.


**Figure 6.5,** interface des images vidéo réelles.

Pour commencer à prendre les images durant le contrôle manuel il faut appuyer sur le bouton "Start", ce qui désactive les boutons "Automatique", "Filtrage" et la saisie de n relatif au retard (voir figure 6.6).

En parallèle, une chaine des paires formées d'images et des vitesses de moteurs successives est créée.

Cette procédure peut être répétée indéfiniment, et chaque fois répétée le nombre des chaînes créées est incrémenté de un et est affiché dans l'interface.

Il faut noter qu'un seul passage sur le chemin ne suffit pas pour entraîner le réseau, il faut effectuer plusieurs passages (entre 10 et 20) pour réussir l'entraînement.

Parfois, il s'avère nécessaire d'éliminer certaines images soit parce qu'elles sont détériorées (bruit), soit parce qu'elles sont inutiles pour l'entrainement (cas où le robot est en arrêt). La suppression de ces images ne devrait pas affecter les autres images de la même chaîne.

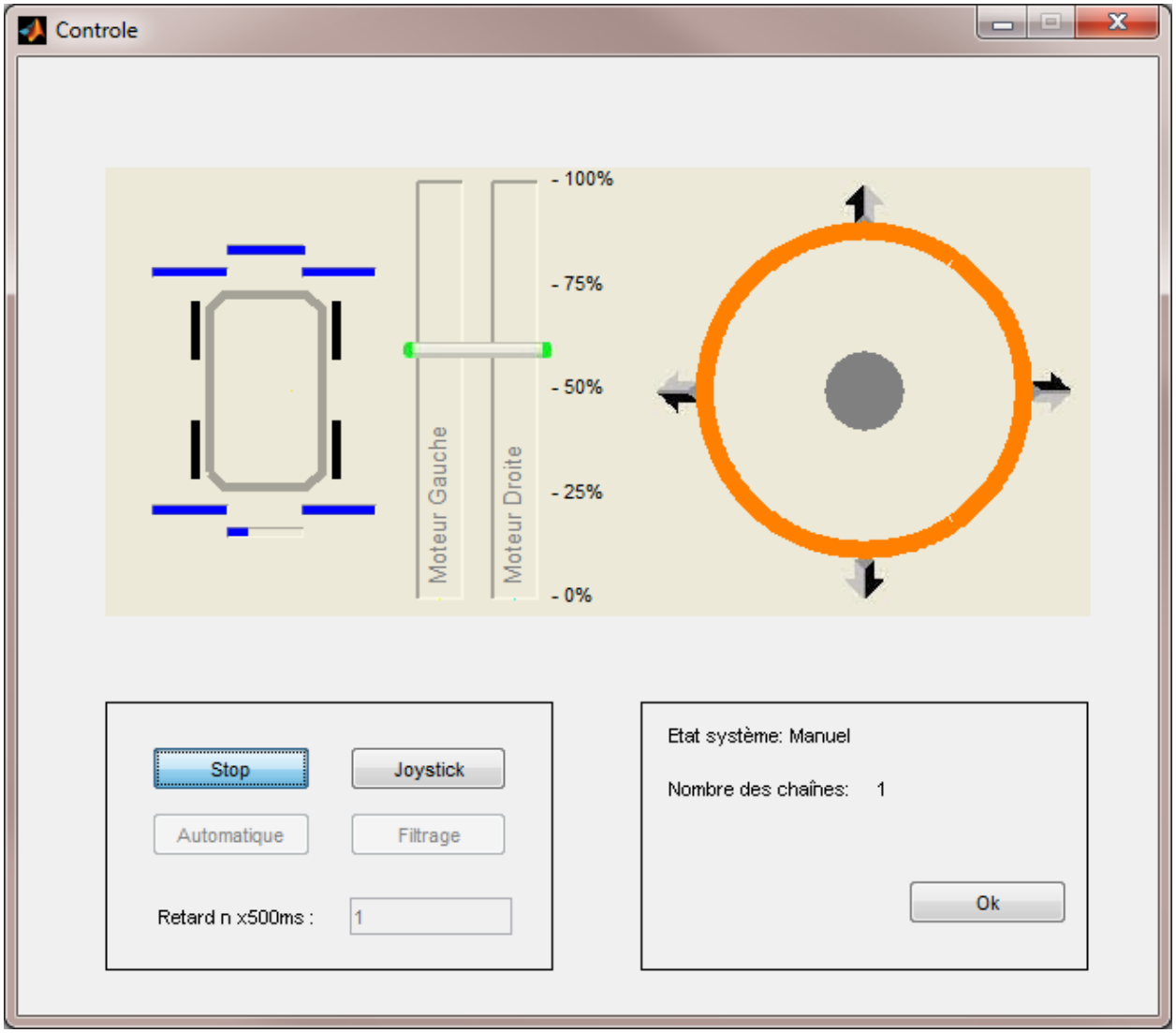



**Figure 6.6,** interface de contrôle en mode manuelle.

Pour faire cette opération de nettoyage, on doit appuyer sur le bouton "Filtrage" pour ouvrir l'interface graphique correspondante (voir figure 6.7).

Cette interface nous aide à manipuler les données acquises (Images et vitesses des moteurs) lors de la phase d'apprentissage.

La barre supérieure nous permet de sélectionner la chaîne à examiner. L'index au-dessous de la barre indique le numéro de la chaîne sur le nombre total des chaînes acquises.

Pour chaque chaîne sélectionnée, la barre inférieure indique le numéro de l'image sélectionnée ainsi que le nombre total d'images de la chaîne.

L'image sélectionnée est visualisée dans la partie gauche de l'interface accompagnée des vitesses des moteurs associées à cette image.

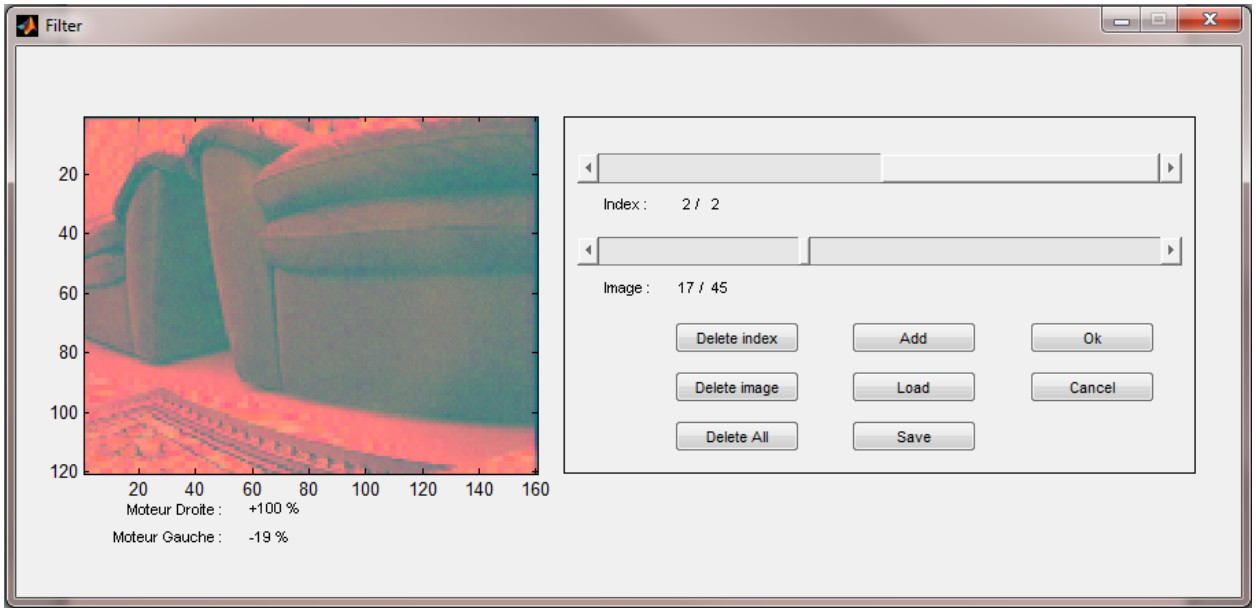



**Figure 6.7,** interface de filtrage.

Le bouton "Delete index" supprime la chaîne sélectionnée et les images dédiées. S'il existe encore une ou plusieurs chaînes disponibles, alors, la chaîne suivante devient active et les deux barres sont initialisées pour celle-là, sinon les barres deviennent inactives.

Le bouton "Delete image", sert à supprimer l'image sélectionnée. Si cette image est la dernière disponible dans la chaîne, alors toute la chaîne est supprimée.

Dans certains cas, l'entrainement peut durer plusieurs heures, voire des jours, d'où l'intérêt de pouvoir sauvegarder les images et données acquises dans un fichier. C'est ce qui est fait lorsqu'on appuie sur le bouton "Save". Une fenêtre de dialogue s'ouvre pour permettre à l'utilisateur de choisir l'emplacement et le nom du fichier de sauvegarde (figure 6.8).

D'autre part, le bouton "Load", nous permet d'ouvrir et de recharger des fichiers sauvegardés précédemment, (figure 6.9).

Le bouton "Add" permet d'ajouter des nouvelles données (images) à un fichier existant.

On peut annuler des modifications déjà faites sur le contenu d'une chaîne ou d'un fichier en appuyant sur le bouton "Cancel".

Aprés avoir choisi les images à utiliser pour l'apprentissage, on ferme les interfaces "Filtrage" et "Control" en pressant sur leurs boutons "Ok" respectifs.

Ensuite, on appuie sur le bouton "Train" de l'interface principal (figure 6.3) pour lancer l'entraînnement du réseau en utilisant les images déjà sélectionnées dans l'étape précédente.

Enfin, pour tester le réseau, on appuie sur le bouton "Automatique" dans l'interface de "Control" (figure 6.4).

**<u>La structure et les principales fonctions du logiciel :</u>**

Le logiciel est composé de plusieurs fichiers écrits en langage Matlab, associés aux interfaces mentionées au dessus. Dans ce paragraphe, nous décrivons brèviemment les principaes fonctions du programme.

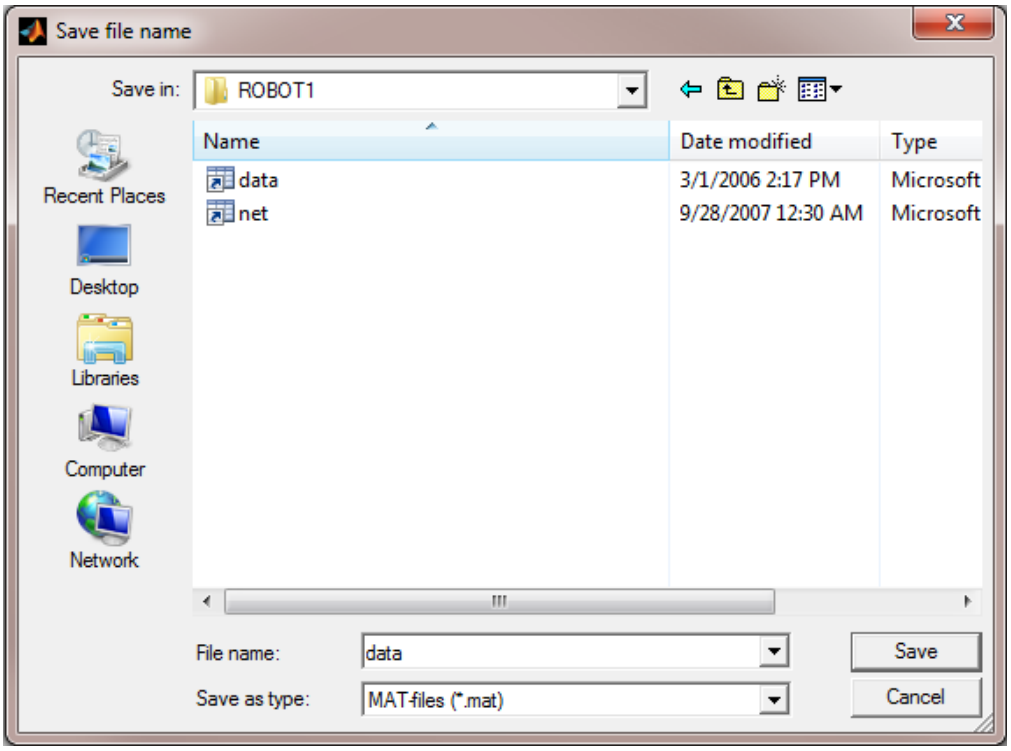



**Figure 6.8,** interface pour sauvegarder les images.

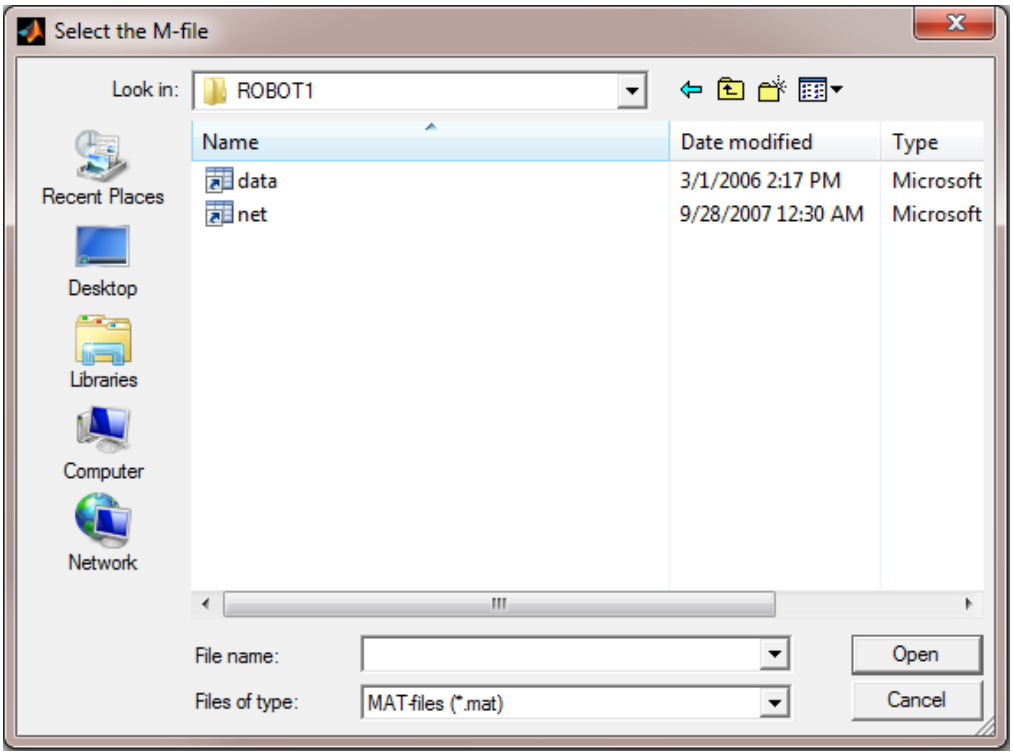



**Figure 6.9,** interface pour ouvrir un fichier.

1. **Les fonctions de l'interface "Main"**
    - Main_OpeningFcn (hObject, eventdata, handles, varargin):
        - Exécutée par Matlab.
        - Rôle:
            a. Afficher l'interface.
            b. Création des variables qui sont utilisées globalement à travers des pointeurs ("handles").
    - pushbutton_connecter_Callback (hObject, eventdata, handles):
        - Exécutée quand le bouton "Connection" est pressé.
        - Rôle:
            a. Lancer l'interface de connexion du port série.
            b. Créer une nouvelle connexion série si le bouton "Ok" de l'interface de connexion est choisi.
            c. Activer le bouton "Control" si le port série est créé.
    - pushbutton_controle_Callback (hObject, eventdata, handles):
        - Exécutée quand le bouton " Control" est pressé.
        - Rôle: Lancer l'interface de contrôle.
    - pushbutton_train_Callback (hObject, eventdata, handles):
        - Exécutée quand le bouton "Train" est pressé.
        - Rôle: Entrainer le réseau de neurones.
    - Serial_Callback (hObject, eventdata):
        - Exécutée quand le port série reçoit 21 octets consécutifs.
        - Rôle:
            a. Mettre à jour les barres d'ultrasons dans l'interface de contrôle.
            b. Arrêter le compteur "ResetJoytim".
            c. Démarrer le compteur "Joytim" (le rôle de ces 2 compteurs est décrit dans l'interface de contrôle).
    - figure1_CloseRequestFcn (hObject, eventdata, handles):
        - Exécutée par Matlab.
        - Rôle:
            a. Détruire l'objet du port série.
            b. Proposer l'option de sauvegarder le réseau.
            c. Fermer l'interface

2. **Les fonctions de l'interface "Control"**
    - Controle_OpeningFcn (hObject, eventdata, handles, varargin):
        - Exécutée par Matlab.
        - Rôle:
            a. Afficher l'interface.
            b. Créer les variables et les drapeaux nécessaires.
            c. Afficher l'interface des images réelles venant du Robot.
            d. Créer 3 compteurs :
                - Le compteur "Joytim" exécute la fonction "Joytimercallback"

100 ms après le début du comptage, il sert à lire la position de la manette au moins chaque 100 ms pour permettre au Robot de répondre avant d'envoyer la commande suivante.

- Le compteur "ResetJoytim" exécute la fonction "ResetJoytimercallback" 500 ms après le début du comptage, il sert à démarrer le compteur "Joytim" si le Robot ne répond pas.
- Le compteur "tim" exécute périodiquement la fonction "timercallback" chaque 100 ms, il est utilisé durant le mode d'acquisition des images.

- axes_b_CreateFcn (hObject, eventdata, handles):
  - Exécutée par Matlab.
  - Rôle: Créer les commandes visuelles de la manette, les barres de vitesse et les barres de la détection ultrasonore.
- Joytimercallback (hObject, eventdata):
  - Exécutée par le compteur "Joytim".
  - Rôle:
    a. En mode manuelle, lire la position de la manette et mettre à jour les commandes visuelles.
    b. En mode automatique, prendre une image instantanée el la traite par le réseau de neurones.
    c. Envoyer les commandes convenables au Robot à travers le port série
    d. Démarrer le compteur "ResetJoytim".
- ResetJoytimercallback (hObject, eventdata):
  - Exécutée par le compteur "ResetJoytim".
  - Rôle: Démarrer le compteur "Joytim".
- timercallback (hObject, eventdata):
  - Exécutée par le compteur "tim".
  - Rôle:
    a. Sauvegarder les images et les vitesses des moteurs dédiées d'une manière périodique durant le mode d'acquisition.
    b. En mode automatique, prendre une image instantanée el la traite par le réseau de neurones.
    c. Envoyer les commandes convenables au Robot à travers le port série
    d. Démarrer le compteur "ResetJoytim".
- Controle_CloseRequestFcn (hObject, eventdata, handles):
  - Exécutée par Matlab.
  - Rôle:
    a. Fermer l'interface des images vidéo réelles et détruire l'objet vidéo.
    b. Détruire les compteurs "tim" et "Joytim".
    c. Fermer l'interface.
- togglebutton_m_Callback (hObject, eventdata, handles):
  - Exécutée quand l'état du bouton à bascule "Automatique" est changé.

- Rôle:
    a. En mode automatique, désactiver les boutons "Filtrage" et "Start" et démarre le compteur "Joytim".
    b. En mode normal, activer les boutons "Filtrage" et "Start".

- togglebutton_s_Callback (hObject, eventdata, handles):
  - Exécutée quand l'état du bouton à bascule "Start" est changé.
  - Rôle:
    a. En phase de démarrage, désactiver les boutons "Filtrage" et "Automatique" et démarrer le compteur "tim".
    b. En phase de croisière, activer les boutons "Filtrage" et "Automatique" et arrêter le compteur "tim".
- pushbutton_f_Callback (hObject, eventdata, handles):
  - Exécutée quand le bouton "Filtrage" est pressé.
  - Rôle: Afficher l'interface "ImFilter".
- togglebutton_ok_Callback (hObject, eventdata, handles):
  - Exécutée quand le bouton "Ok" est pressé.
  - Rôle: Fermer l'interface.
- pushbutton_r_Callback (hObject, eventdata, handles):
  - Exécutée quand le bouton " Joystick" est pressé.
  - Rôle: Démarrer le compteur "Joytim".

**3. Les fonctions de l'interface "ImFilter"**

- ImFilter_OpeningFcn (hObject, eventdata, handles, varargin):
  - Exécutée par Matlab.
  - Rôle: Initialiser les barres de sélection "index" et "image".
- pushbutton_dindex_Callback (hObject, eventdata, handles):
  - Exécutée quand le bouton "Delete index" est pressé.
  - Rôle:
    a. Suprimer la série des images indiquée par la barre "index".
    b. Mettre à jour les barres de sélection "index" et "image".
- pushbutton_l_Callback (hObject, eventdata, handles):
  - Exécutée quand le bouton "Load" est pressé.
  - Rôle:
    a. Offrir l'option de sauvegarder les nouvelles images.
    b. Exécuter la fonction "load_cell".
- pushbutton_d_Callback (hObject, eventdata, handles):
  - Exécutée quand le bouton "Delete All" est pressé.
  - Rôle:
    a. Supprimer toutes les images.
    b. Mettre à jour les barres de sélection "index" et "image".
- pushbutton_dim_Callback (hObject, eventdata, handles):
  - Exécutée quand le bouton "Delete image" est pressé.
  - Rôle:

        a. Supprimer l’image idiquée par la barre "image".
        b. Mettre à jour les barres de sélection "index" et "image".
- pushbutton_s_Callback (hObject, eventdata, handles):
    - Exécutée quand le bouton "Save" est pressé.
    - Rôle: Exécuter la fonction "save_cell".
- pushbutton_add_Callback (hObject, eventdata, handles):
    - Exécutée quand le bouton "Add" est pressé.
    - Rôle:
        a. Offrir l’option de sauvegarder les images courantes.
        b. Exécuter la fonction "add_cell".
- pushbutton_ok_Callback (hObject, eventdata, handles):
    - Exécutée quand le bouton "Ok" est pressé.
    - Rôle:
        a. Fermer l’interface.
        b. Passer un pointeur à l’interface de contrôle pour les images modifiées.
- pushbutton_cancel_Callback (hObject, eventdata, handles):
    - Exécutée quand le bouton "Cancel" est pressé.
    - Rôle: Fermer l’interface.
- slider_index_Callback (hObject, eventdata, handles):
    - Exécutée quand la position de la barre "index" est changée.
    - Rôle: Mettre à jour les barres de sélection "index" et "image".
- slider_im_Callback (hObject, eventdata, handles):
    - Exécutée quand la position de la barre "image" est changée.
    - Rôle: Mettre à jour la barre de sélection "image".
- apdateindexsel_slider (handles, index):
    - Exécutée par d’autres fonctions pour mettre à jour la barre "index".
    - Rôle:
        a. Mettre à jour la barre de sélection "index".
        b. Afficher la position de la barre dans le texte "index".
- apdateimagesel_slider (handles, index, im):
    - Exécutée par d’autres fonctions pour mettre à jour la barre "image".
    - Rôle:
        a. Mettre à jour la barre de sélection "index".
        b. Afficher la position de la barre dans le texte "image".
        c. Afficher l’image convenable sur l’interface.
- load_cell (handles):
    - Exécutée par d’autres fonctions pour ouvrir un nouveau fichier d’images.
    - Rôle:
        a. Afficher une interface pour ouvrir un nouveau fichier d’image.
        b. Mettre à jour les barres de sélection "index" et "image".
- add_cell (handles):
    - Exécutée par d’autres fonctions pour ajouter un nouveau fichier d’images.

- Rôle:
    a. Afficher une interface pour ajouter un nouveau fichier d'images aux images existantes.
    b. Mettre à jour les barres de sélection "index" et "image".

- save_cell (handles):
    - Exécutée par d'autres fonctions pour sauvegarder les images dans un fichier.
    - Rôle: Afficher une interface pour sauvegarder les images dans un fichier

## *6.4 Conclusion:*

Dans ce chapitre, nous avons présenté le logiciel qui nous permet d'entraîner et de tester le réseau neuronal que nous utilisons pour contrôler le mouvement du robot d'une manière autonome.

Notons que ce logiciel, bien qu'il est conçu pour cette application, il est toutefois possible de le modifier, sous Matlab, afin de l'adapter à d'autres applications ou pour l'enrichir en lui ajoutant des outils supplémentaires.

# PARTIE III

## Chapitre 7 TESTS ET EVALUATION

# Chapitre 7 Tests et Évaluation

## 7.1 Introduction:

A ce moment nous avons tous les ingrédients, c'est le temps de mettre notre réseau en test. Dans ce chapitre nous allons mentionner étape par étape les phases d'acquisition et d'apprentissage. Un paragraphe sera ensuite dédié aux observations sur lesquelles sera basée le choix définitif de l'architecture du réseau.

Dans le chapitre précédent nous n'avons pas mentionné les fonctions d'activation et d'apprentissage utilisées; dans ce chapitre, nous testons et validons le choix de ces fonctions.

## 7.2 Les algorithmes d'entraînement disponibles sous Matlab:

La figure 7.1 montre le parcours et l'environnement où le robot est testé. Comme la figure indique, le robot débutera de la partie gauche de la figure en suivant la flèche 1. Lorsqu'il atteint la partie droite il tournera de 90° vers la gauche. Ensuite il avance vers la partie supérieure en suivant la flèche 2, quand il atteint la destination il s'arrête. Le réseau doit naviguer le robot d'une manière autonome en se basant sur l'analyse d'images de taille de 60x80 pixels acquises par la caméra montée au bord du robot.

La première étape est de naviguer le robot manuellement. Durant cette étape, les images réelles acquises constituent les informations présentées à l'entrée du réseau pour l'entrainer. On a répété cette étape 6 fois, donc on a 6 chaines ou séries d'images qui correspondent au chemin parcouru.

Avant de commencer la phase d'apprentissage, il fallait éliminer les images indésirables en utilisant le "GUI Filtrage", ce filtrage est essentiel pour la pertinence des décisions prises par le réseau.

Pour effectuer la phase d'apprentissage, on s'est servi de plusieurs fonctions d'apprentissage disponibles sous MATLAB.

La mesure de performances est effectuée en utilisant la fonction MSE (Mean Squared Normalized Error, la Moyenne normalisée des carrés des écarts) prédéfinie dans Matlab. Les écarts sont calculés par la différence entre les valeurs des sorties désirées et celles des sorties obtenues. Cette fonction de mesure des performances permet de déterminer si le réseau à entrainer a atteint la performance voulue.

La fonction MSE n'est pas la seule fonction utilisée pour mesurer la performance mais elle est choisie par défaut par les fonctions d'apprentissage utilisées.

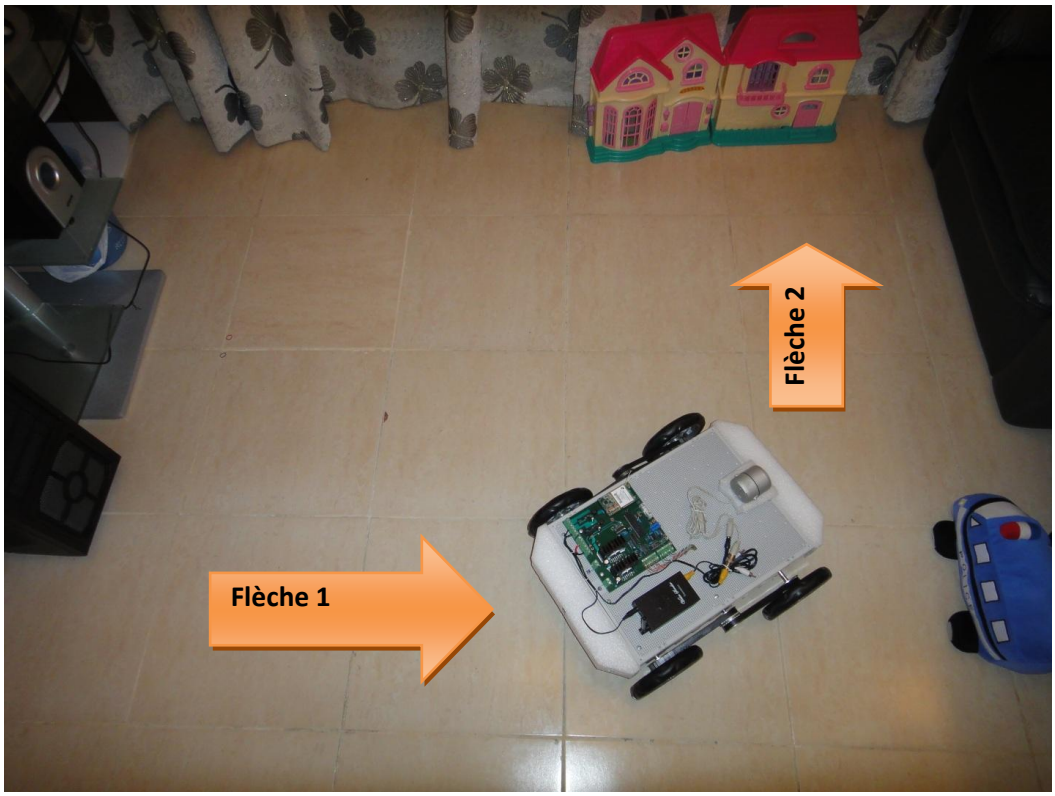



**Figure 7.1,** le parcours et l'environnement où le robot est testé.

D'après MATLAB, l'algorithme de "Levenberg-Marquardt" (fonction trainlm) a la plus rapide convergence vers la solution pour les petits réseaux, surtout lorsqu'il s'agit d'un problème d'approximation des fonctions. Cette fonction d'apprentissage (trainlm) permet d'obtenir une MSE minimale, ce qui correspond à un réseau bien entraîné et donc plus précis.

Par contre, pour des grands réseaux tels que ceux destinés à la reconnaissance des modèles et des formes (comme dans notre cas) la performance de "trainlm" est médiocre, et en plus, elle est assez gourmande en mémoire.

D'autre part, l'algorithme de "Resilient Backpropagation" (fonction trainrp) converge plus rapidement en cas de la reconnaissance des formes et exige peu de mémoire. Par contre, il n'est pas assez performant lorsqu'il s'agit d'approximation d'une fonction ou lorsqu'on cherche à obtenir une MSE minimale.

L'algorithme de "conjugate gradient" (particulièrement la fonction trainscg) est performant presque dans tous les problèmes. Sa convergence est aussi rapide que les 2 premiers, que ce soit pour l'approximation d'une fonction ou pour la reconnaissance des formes. En plus, cette convergence est plus rapide pour les grands réseaux.

L'algorithme de "quasi-Newton Backpropagation" (fonction trainbfg) est similaire que trainlm mais exige moins de mémoire que cette dernière. Or, son coût de calcul augmente fortement lorsque la taille du réseau devient importante.

Enfin, l'algorithme "One Step Secant" (fonction trainoss) constitue un compromis entre "trainbfg" et

"trainscg", en ce qui concerne la vitesse de convergence et le besoin en mémoire.

Pour confirmer les caractéristiques des algorithmes d'entraînement cités ci-dessus, nous avons entraîné notre réseau en utilisant chacune des fonctions d'entraînement déjà citées, et ce en se servant d'une série choisie dans l'ensemble d'apprentissage constitué de six séries d'images capturées.

Les résultats obtenus sont présentés au tableau 7.1. Les fonctions "trainlm" et "trainbfg" ont épuisé toute la mémoire disponible, avant même de compléter la première itération, ce qui a entraîné l'échec du processus d'apprentissage.

La fonction "trainrp" a effectué le plus grand nombre d'itérations (1000), mais la MSE n'a pas passé au-dessous de 0.1 (figure 7.2).

Le tableau montre que les fonctions "trainscg" et "trainoss" ont réalisé les meilleures performances (figures 7.3 et 7.4). Comme le tableau indique aussi, la durée d'une itération de "trainoss" est plus courte que celle de "trainscg", mais cette dernière converge plus rapidement avec un nombre d'itérations plus petit. Cette comparaison montre sans doute que le choix le plus convenable est la fonction "trainscg" de tout point de vue.

**Tableau 7.1, Résultats d'exécution des différentes fonctions d'apprentissage utilisées.**

| Fonction | MSE | Nombre d'itérations | Durée d'exécution(h:mn:s) | Durée d'une itération (s) |
|---|---|---|---|---|
| trainlm | - | - | - | - |
| trainbfg | - | - | - | - |
| trainrp | .117 | 1000 | 1:24:13 | 5 |
| trainscg | .01 | 113 | 24:41 | 13,1 |
| trainoss | .01 | 184 | 26:24 | 8,6 |

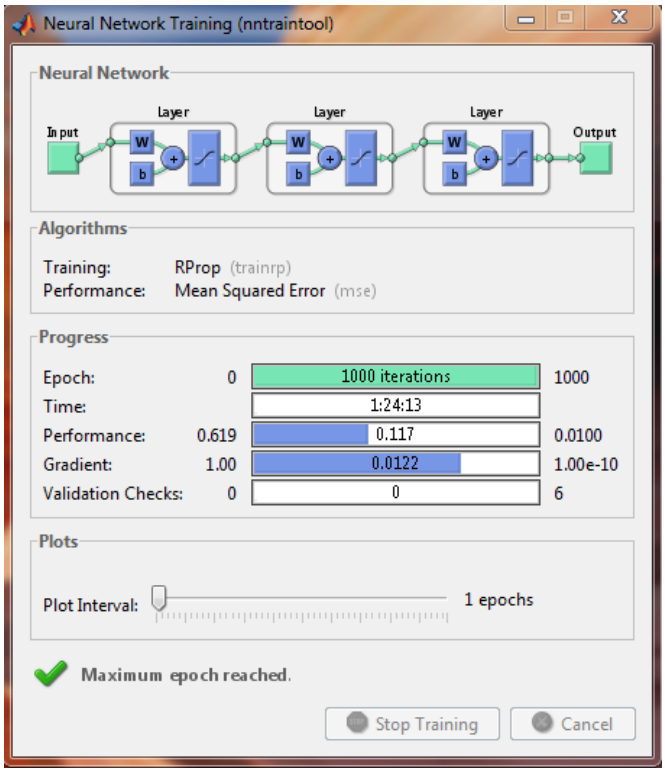



**Figure 7.2,** État d'avancement de l'exécution de trainrp.

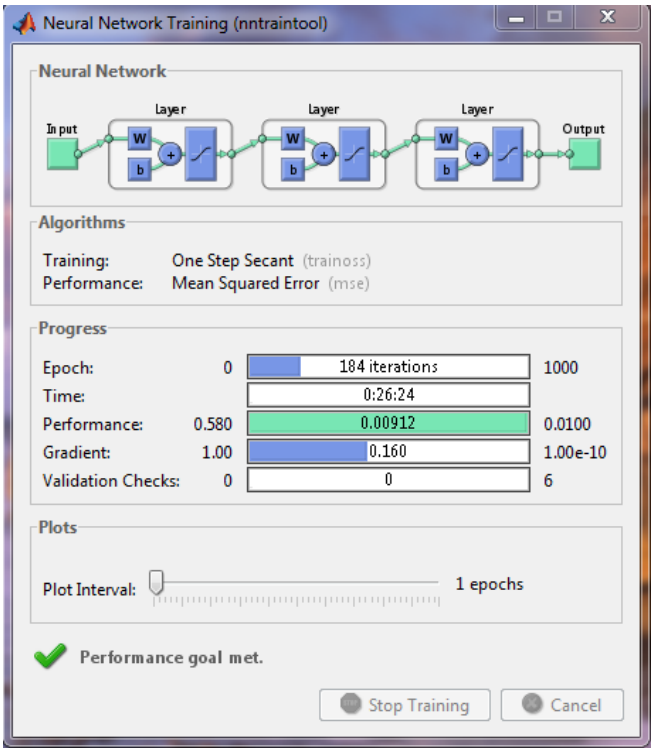



**Figure 7.3,** État d'avancement de l'exécution de trainoss.

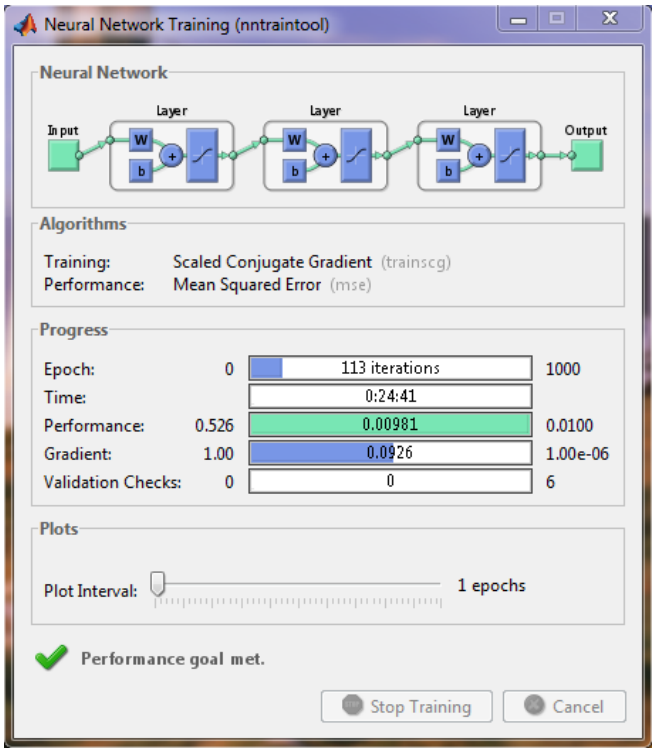



**Figure 7.4,** État d'avancement de l'exécution de trainscg.

## *7.3 La phase d'apprentissage:*

Dans la phase d'apprentissage, nous avons effectué l'entraînement d'une manière progressive: nous entraînons le réseau en utilisant une des six séries d'images, puis en utilisant deux de ces six, puis trois et ainsi de suite jusqu'à l'utilisation de toutes les séries en même temps.

Si l'entraînement est fait en séparant les séries, le réseau s'adapte à la dernière série seulement. Le tableau 7.2 montre la progression de l'entraînement.

**Tableau 7.2, Résultats obtenus lors de l'apprentissage progressif.**

| Nombre de séries | Nombre Total d'images | Nombre d'itérations | MSE | MSE au départ | Durée d'exécution (h:mn:s) | Durée d'une itération (s) |
|---|---|---|---|---|---|---|
| 1 | 32 | 113 | 0,01 | 0,526 | 24:41 | 13,1 |
| 2 | 75 | 135 | 0,01 | 0,0443 | 1:13:00 | 32,44 |
| 3 | 114 | 145 | 0,01 | 0,0289 | 2:18:06 | 57,14 |
| 4 | 162 | 156 | 0,01 | 0,025 | 4:45:11 | 109,68 |
| 5 | 216 | 172 | 0,01 | 0,0231 | 9:46:27 | 204,57 |
| 6 | 255 | 1000 | 0,0166 | 0,0687 | 71:51:37 | 258.7 |
| 7 | 406 | 500 | 0,0341 | 0,0843 | 192:13:20 | 1384 |

Après la phase d'apprentissage, nous avons mis en test le réseau entraîné. Le réseau a réussi 5 de 10 tentatives donc 50% de réussite. Le réseau a failli 5 fois de faire tourner de 90° le robot. Pour améliorer les performances de l'entraînement, nous avons ajouté une septième série et on a entrainé le réseau de nouveau, le nombre total des images utilisées est passé à 406. L'entraînement a mis plus de 8 de jours pour effectuer 500 itérations seulement (la durée d'une itération a dépassé les 23 mn) et le niveau de

MSE atteint n'a pas été meilleur de celui réalisé avec les 6 séries d'images (0,0341 contre 0,0166 auparavant)!

Malgré cela, le réseau a réussi 9 de 10 tentatives soit un taux de 90% de réussite. Il parait que, chaque fois on ajoute une série d'images, le réseau tend à mieux se généraliser, bien qu'on n'a pas pu atteindre une MSE plus faible durant la phase d'apprentissage.

## *7.4 Quelques Observations:*

Dans le chapitre, nous avons mentionné quelques fonctions d'activation. Pour notre application, nous nous sommes intéressés à la fonction d'activation "sigmoïde tangente hyperbolique". Cette fonction introduit la non-linéarité du réseau et compresse la sortie dans un intervalle entre -1 et +1.

**Observation 1 :**

Selon notre conception du réseau, la sortie est activée si elle est égale à +1, sinon elle est égale à -1. En utilisant la fonction d'activation mentionnée ci-dessus, et en remplaçant -1 par 0, le réseau tend à s'entraîner plus rapidement ; or, dans ce cas, l'écart entre les deux états devient plus court, ce qui réduirait les performances du réseau.

Par exemple, lorsque l'état désactivé correspond à 0 et que l'erreur du réseau entraîné est égale à 0,1 pour une sortie activée, alors l'écart entre les deux états sera, au minimum, égal à 0,9, alors que si l'état désactivé correspond à -1, on assure un écart minimal égal à 1,9, ce qui résulte en un réseau plus fiable.

**Observation 2:**

Sans aucun doute, si on incrémente le nombre des neurones dans les couches constituant le réseau, ce dernier tend à mieux se généraliser et à résoudre des problèmes plus complexes. D'autre part cet agrandissement du réseau conduit à un calcul plus lourd donc à une période d'apprentissage plus longue.

**Observation 3 :**

Les images utilisées pour l'apprentissage sont de type "grayscale" (luminance seule) et de dimensions 60x80 pixels (Figures 7.5). La qualité de ces images parait suffisante pour nos tests.

Toutefois, pour des applications réelles cette résolution peut s'avérer insuffisante; une meilleure résolution (Figure 7.6), voire une information sur les couleurs (Figure 7.7), pourraient être exigées.

Or, améliorer la résolution ou utiliser des images en couleur exige l'augmentation du nombre des neurones dans le réseau et donc les poids synaptiques, ce qui conduirait certainement à une phase d'apprentissage plus lourde.

**Observation 4 :**

Pour que le réseau soit capable de se généraliser et de donner les résultats désirés, il faut l'alimenter d'une manière périodique par des nouvelles informations (séries des images) qui les utilise pour

améliorer son entraînement. Ces nouvelles images doivent être concaténées avec les précédentes, sinon, le réseau s'adapte aux nouvelles sans tenir compte des anciennes.

**Observation 5 :**

L'habilité du réseau neuronal à se généraliser lui permet de négliger des données contenues dans les images sans altérer les décisions prises. On a testé ceci, en modifiant certains détails d'une même scène filmée dans deux séries d'images différentes (par exemple, la présence d'une chaise sur le canapé (figure 7.8) et son absence (figure 7.9)). Cette modification n'a pas altéré la qualité de l'entraînement.

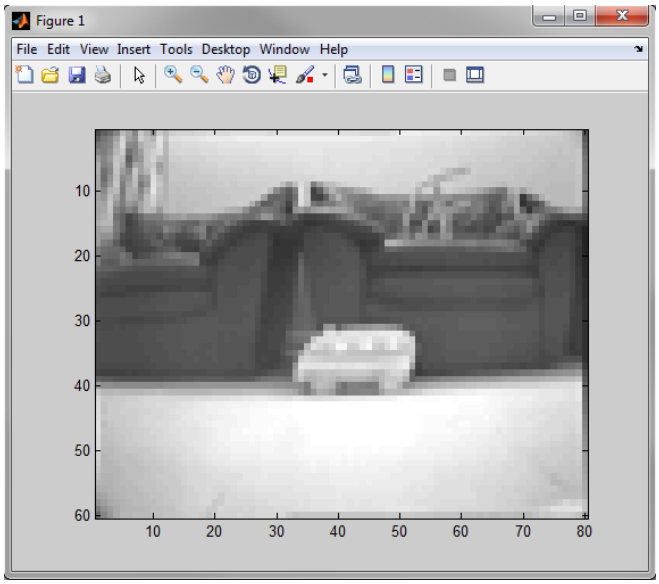



**Figure 7.5,** image 60x80 pixels en grayscale.


**Figure 7.6,** image 120x160 pixels en grayscale.


**Figure 7.7,** image 120x160 pixels en couleur.

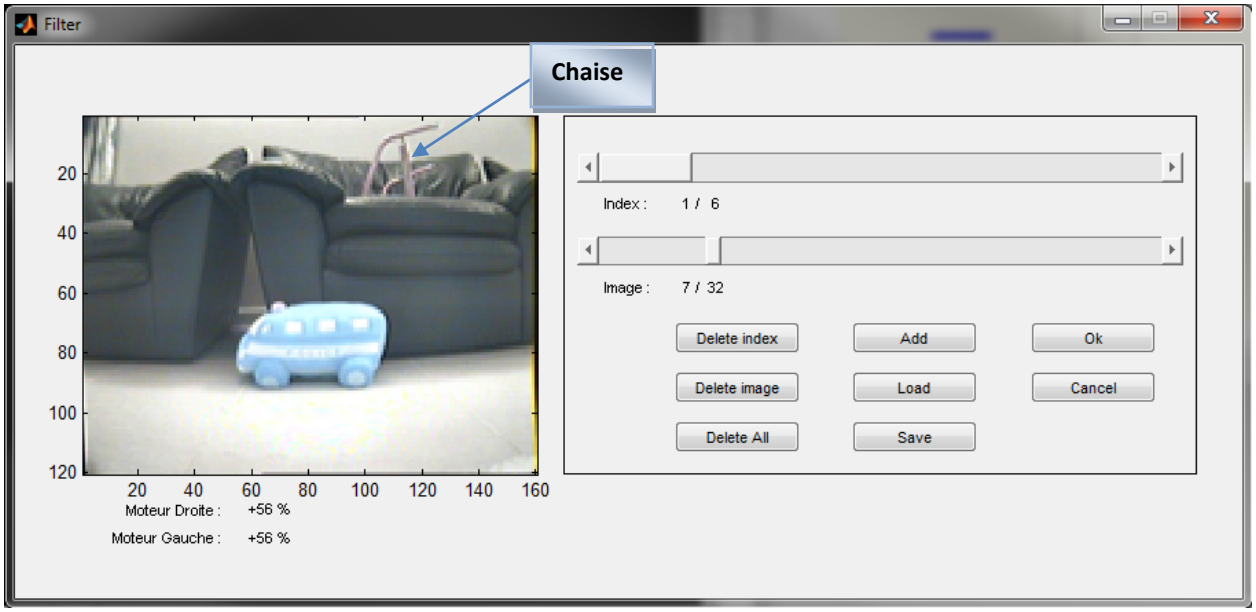



**Figure 7.8,** la figure indique la présence d'une chaise dans la chaine d'images no. 1.

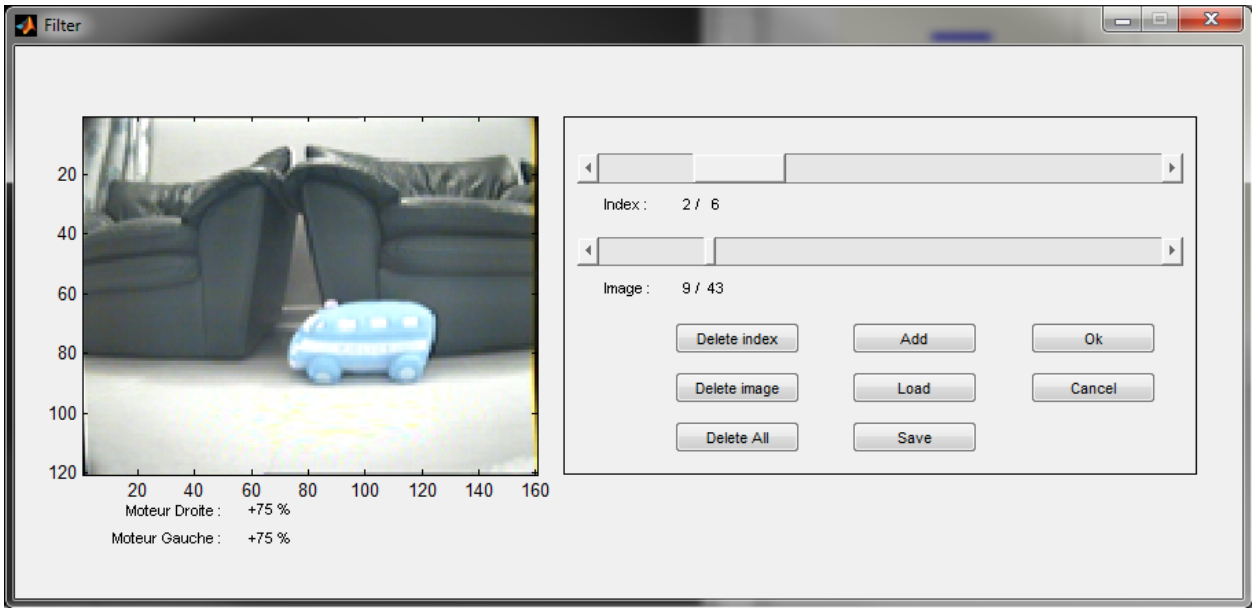



**Figure 7.9,** la figure indique l'absence de la chaise dans la chaine d'images no. 2.

D'après ces observations, on peut constater que la réalisation d'un réseau capable de naviguer un robot mobile dans une surface simple, prédéfinie et limitée est faisable, malgré cela, pour mettre cela dans une application réelle comme la navigation d'une voiture exige l'utilisation de plusieurs processeurs engagés dans le calcul en parallèles et une mémoire capable de porter toutes les informations nécessaires pour la phase d'apprentissage, ce qui rend l'application beaucoup trop coûteuse.

De même, comme indiqué dans l'observation 5, le réseau a prouvé qu'il est aveugle aux objets qui l'interceptent, alors l'utilisation du réseau neuronal dans une application réelle requiert l'intégration d'un système d'évitement des obstacles avec le réseau.

## 7.5 Conclusion:

En vue des essais et les observations décrits ci-dessus nous pouvons conclus que :

- Pour notre réseau, la meilleure fonction d'apprentissage est trainscg. Cette fonction a les meilleures performances pour un réseau de neurones utilisé pour la classification de modèle avec un grand nombre de poids synaptique.
- L'apprentissage du réseau est une procédure continue, il fallait toujours acquérir des nouvelles séries des images pour l'entrainer afin de réduire les décisions fautes du réseau. Bien sûr cela augmente considérablement la taille du mémoire utiliser et nécessite l'utilisation d'un ordinateur bien plus rapide.
- D'après l'observation 5 le réseau tend à généraliser les décisions, donc de négliger des détails importants dans les images comme un obstacle, pour cela la navigation par un réseau de neurone doit être associée à un system de détection d'obstacle similaire à celui utilisé.
- Malgré la capacité d'un réseau neuronal de naviguer un robot mobile dans un environnement limité, actuellement la technologie existante ne nous permet pas la réalisation d'un réseau pratique et économique capable de naviguer un moyen de transport d'une manière autonome en comptant sur des images réelles.

La conclusion ci-dessus ne signifie pas qu'un réseau neuronal est incapable de naviguer dans un milieu plus complexe en utilisant la technologie valable. Si à la place des informations sous forme des images on utilise par exemple les informations venant d'un capteur GPS (Global Positioning System) et on l'associe à un détecteur des obstacles, la réalisation pratique et économique du système sera plus vérifiable.

D'autre part, on peut naviguer le robot en utilisant une différente approche, par exemple un ordinateur qui contienne un plan de navigation peut contrôler le robot en lui envoyant les coordinations de GPS en tenant compte des informations obtenues à l'aide d'un capteur GPS monté sur le robot, mais c'est un sujet pour un différend projet.

# Annexe 1 HISTORIQUE DES ROBOTS.

Les premiers pionniers dans le domaine de robotique sont :

L'inventeur et le mathématicien Grec Ctesibius (285–222 BC) qui est reconnu comme le "père des pneumatiques". Il est le créateur du premier orgue à eau et l'horloge à eau nommés clepsydra(voir figure 1.1).

L'ingénieur, l'auteur et le mécanicien Philo de Byzance (280–220 BC), connu aussi par Philo Mechanicus, est le créateur du premier lavabo mécanique (voir figure 1.2).

Le mathématicien et l'ingénieur Grec (voir figure 1.3) Heron de Alexandria (10–70 AD), fut l'inventeur de l'Aeolipile, la première machine à vapeur et la première machine de vendre (!!!???).

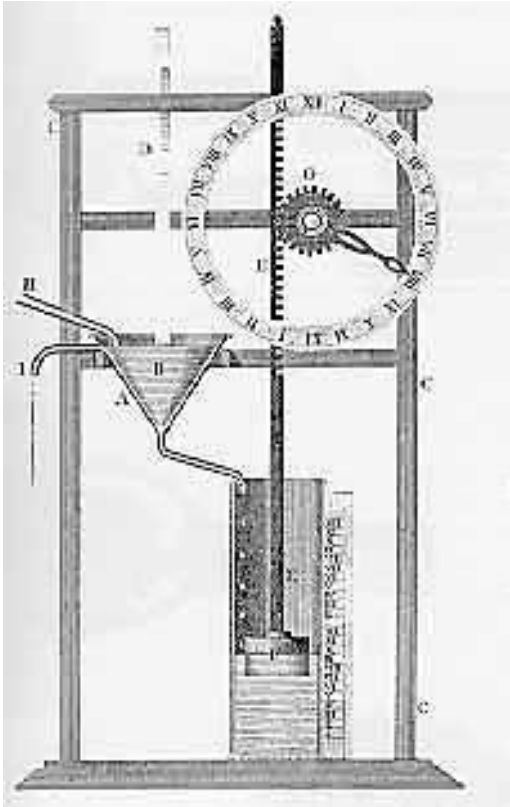


**Figure 1.1**, Horloge Clepsydra créé par Ctesibius.

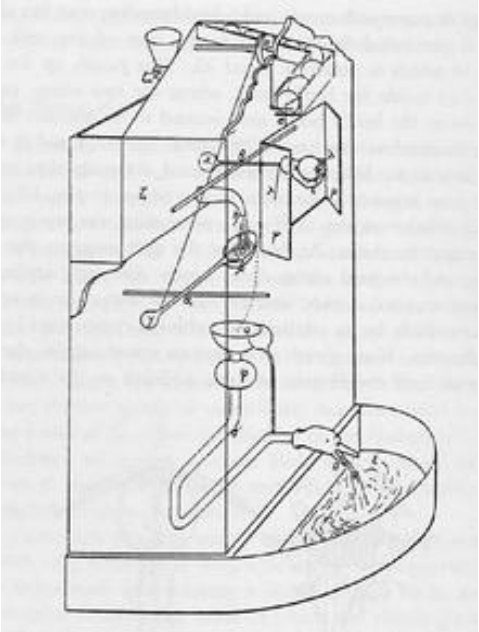


**Figure 1.2**, Lavabo mécanique créé par Philo.

**Figure 1.3**, Héron de Alexandria (10–70 AD).

Al-Jazari (1136–1206) est un inventeur Musulman qui avait créé un robot musical programmable capable de remplacer 4 musiciens.

Vers le milieu du deuxième millénium, des machines encore plus complexes furent conçues:

Leonardo da Vinci (1452–1519) a conçu un robot humanoïde (voir figure 1.4) qui est un chevalier mécanique capable de s'assoir, de saluer, de bouger sa tête et sa mâchoire.

En 1739 un Canard Digérateur est construit par Jacques de Vaucanson (voir figure 1.5). Ce canard mange des grains et fictivement les métabolise et les déféque.

Hisashige Tanaka (1799–1881) a inventé une série des jeux de mécanique très complexes, qui pouvaient servir de thé, tirer des flèches et peindre.

Vers la fin du XIX siècle, et durant l'époque industrielle, les techniques mécaniques se sont nettement progressées. Ainsi, on trouvait des réalisations pratiques comme celle de Nikola Tesla, une torpille télé-contrôlée à travers une liaison hertzienne en 1898.

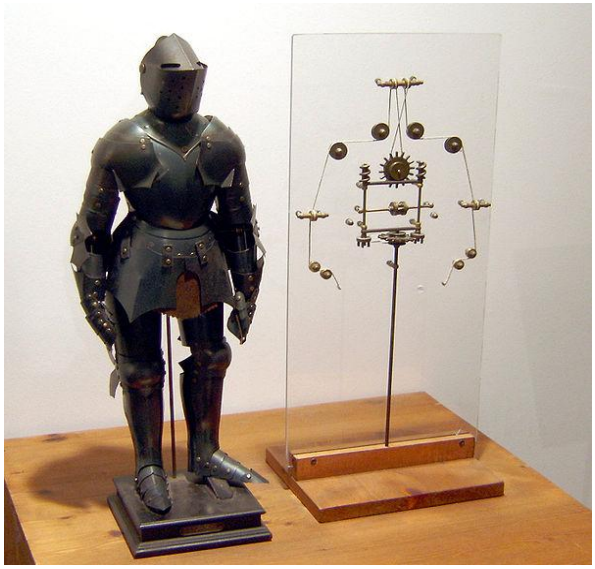


**Figure 1.4**, Un prototype du chevalier mécanique (Robot de Leonardo da Vinci).

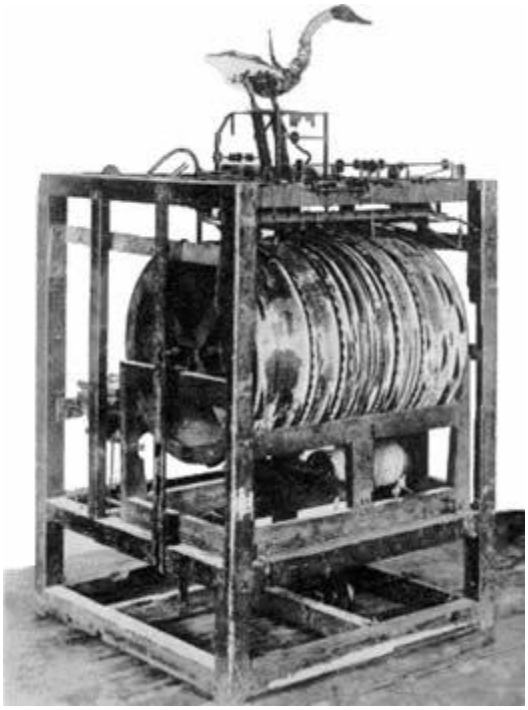


**Figure 1.5**, Un canard digérateur de Jacques de Vaucanson.

Au-delà nous trouvons de plus en plus des androïdes dont le but est d'imiter les caractéristiques humaines, comme la création du biologiste Makoto Nishimura « Gakutensoku » en 1929, qui est capable de pleurer et de montrer des expressions humaines ; de même, la création de Westinghouse « Electro » en 1938 (figure 1.6).

Depuis quelques décennies, l'électronique est devenue la base de contrôle pour les robots à la place de la mécanique. Le premier robot autonome contrôlé électroniquement était « Elmer et Elsie », nommé aussi la tortue robotique, (figure 1.7), créé par William Grey Walter en 1948. Ce robot est capable de trouver d'une manière autonome la station dédiée au chargement de ses propres batteries quand ces dernières l'en avaient besoin.

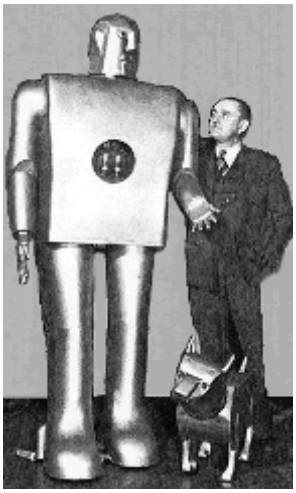


**Figure 1.6**, Electro, l'androïde de Westinghouse.

**Figure 1.7**, La tortue robotique.

Le premier robot digital et programmable "Unimate" (figure 1.8), est créé en 1956 par George Devol. La première utilisation industrielle de ce bras robotique était en 1961 par General Motors.

Depuis ce temps, la technologie des robots s'est mûrie et elle constituait, enfin, une véritable assimilation des différentes technologies comme le robot ASIMO (figure 1.9), qui est capable de bouger et marcher comme l'être humain.

Actuellement, les robots peuvent être classés dans deux grandes catégories:

1. Les robots utilisés dans la Recherche comme les humanoïdes (ex : ASIMO) et comme les Nano

robots.

2. Les robots commerciaux et industriels, capables d'exercer des tâches bien déterminées et d'une manière nettement plus précise et plus fiable que celle de l'être humain. De plus, ils peuvent être utilisés dans des milieux dangereux où les conditions sont inappropriées pour l'homme. Ces robots sont principalement utilisés dans les usines, le transport, l'exploration de l'espace, les opérations chirurgicales, l'armement, et les laboratoires de recherche.

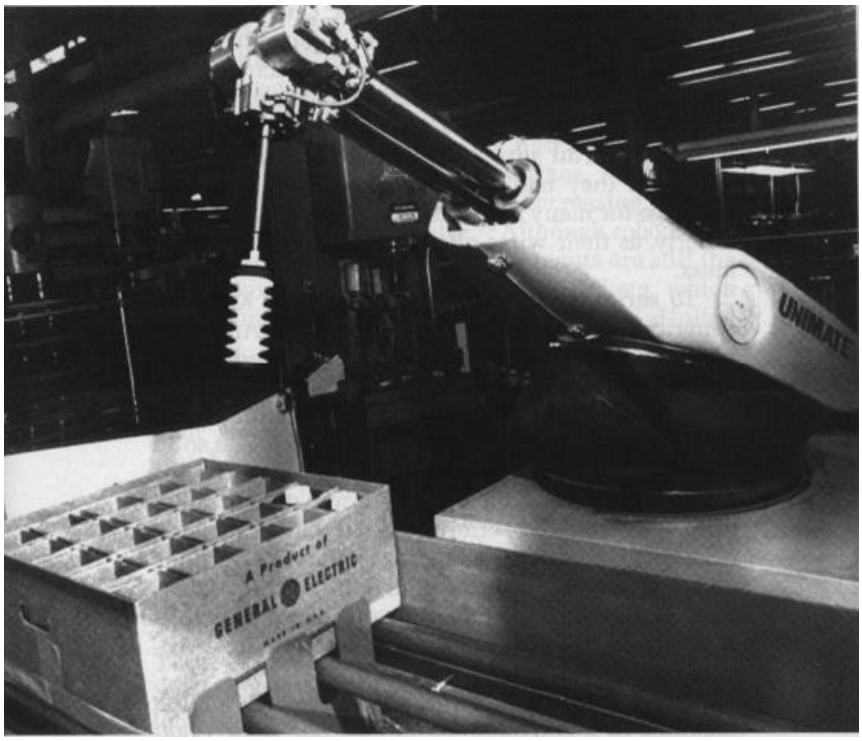



**Figure 1.8**, Le bras robotique Ultimate.

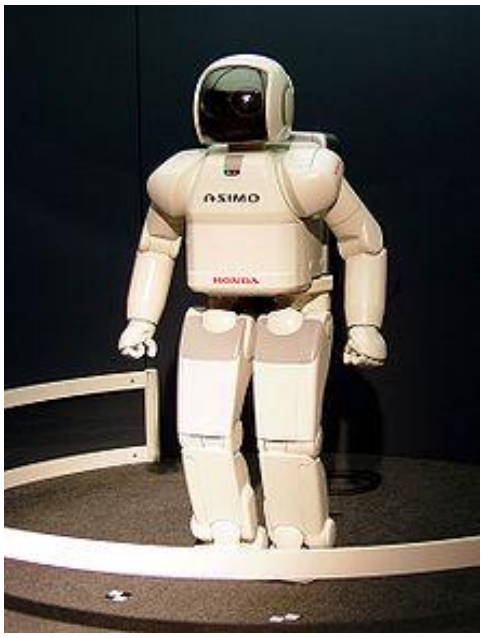



**Figure 1.9**, L'humanoïde ASIMO du Honda.

# ANNEXE 2

**Schéma 1. Circuit électronique du contrôleur.**
**Schéma 2. Placement des composantes du contrôleur sur le PCB.**
**Schéma 3. Circuit électronique de l'interface PC.**
**Schéma 4. Placement des composantes de l'interface PC sur le PCB.**

**Schéma 1.1**

neuron_robot_driver.sch-1 - Fri Feb 05 08:53:58 2010

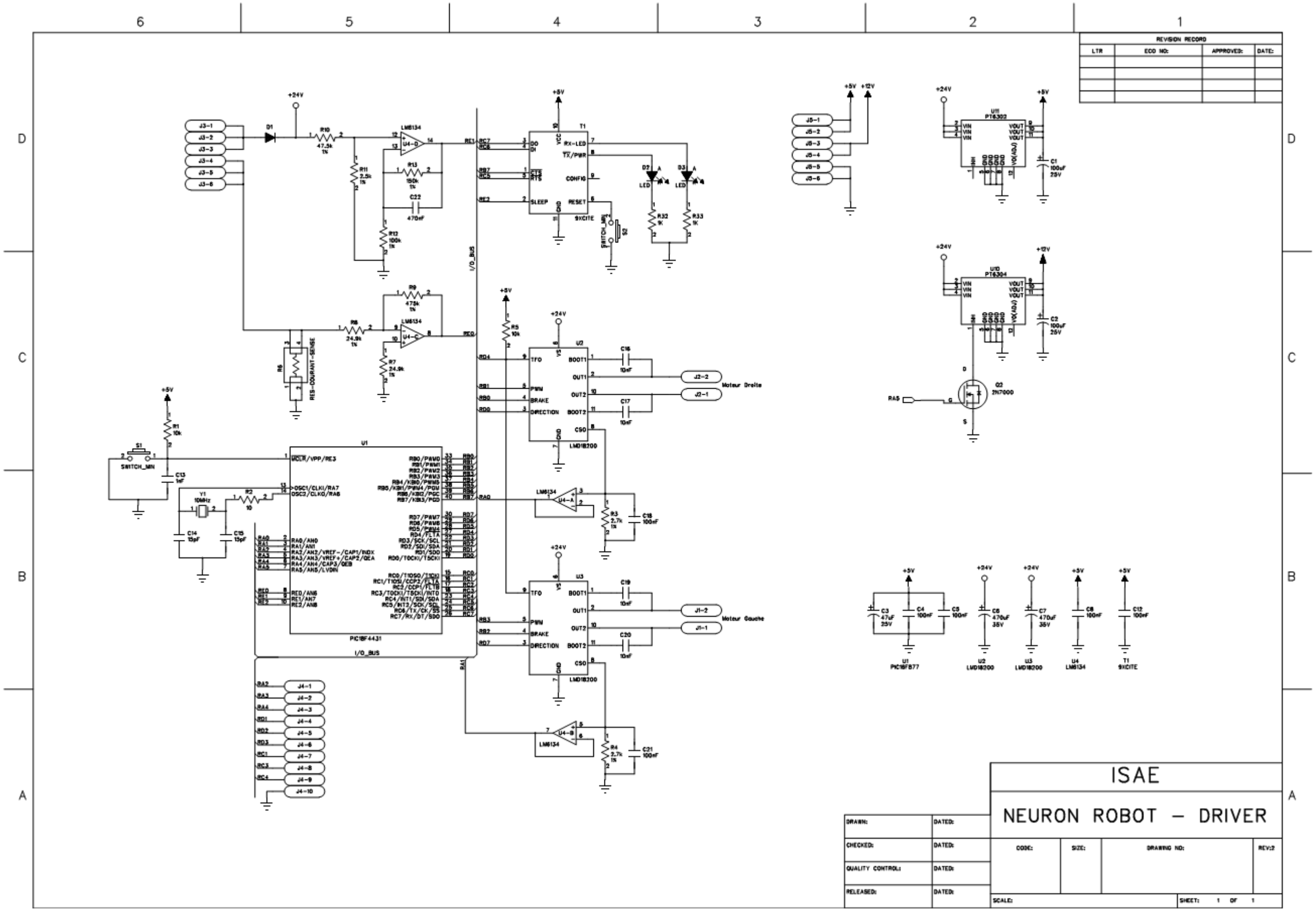
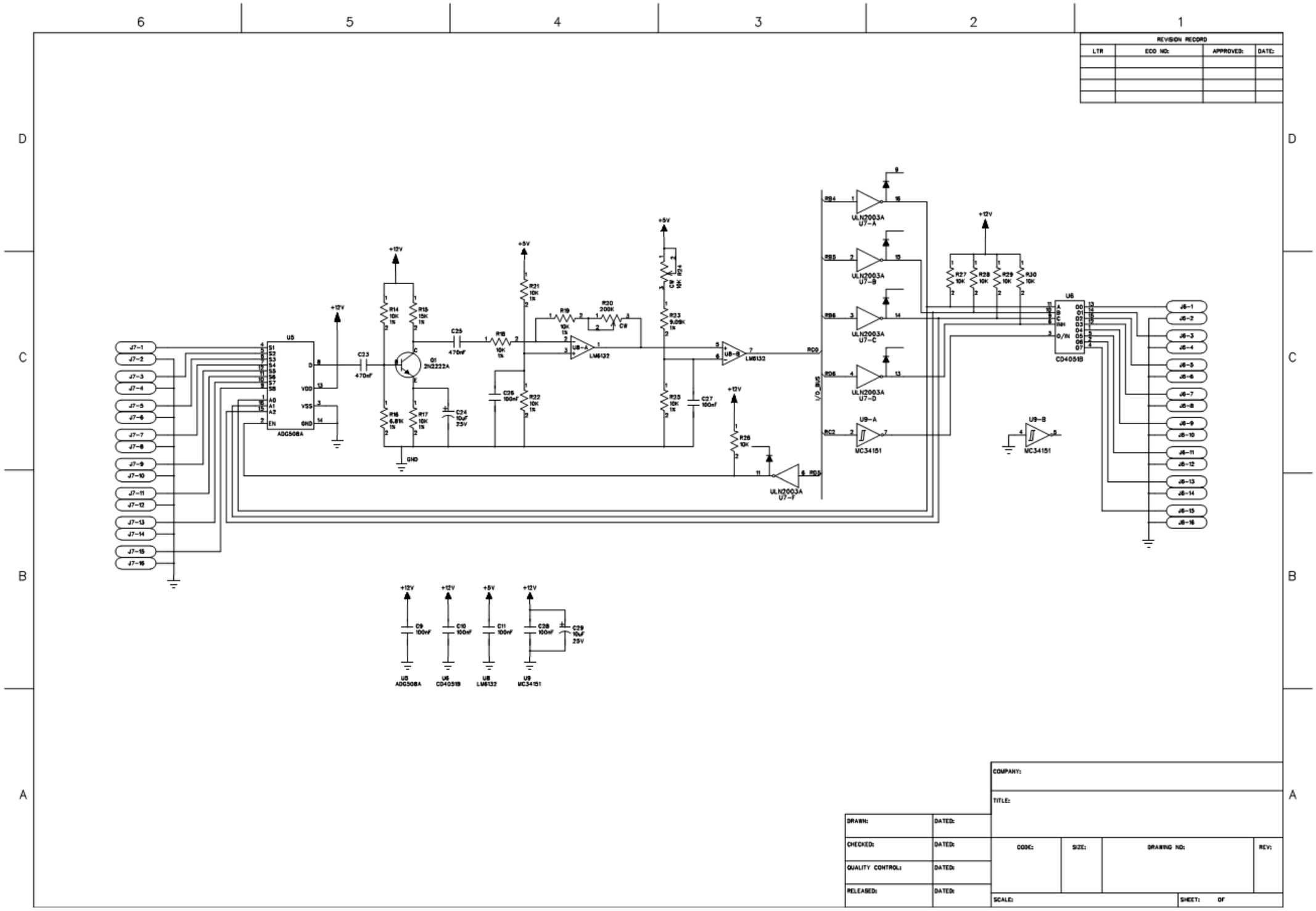


**Schéma 1.2**

neuron_robot_driver.sch-2 - Fri Feb 05 08:53:58 2010



## Schéma 2

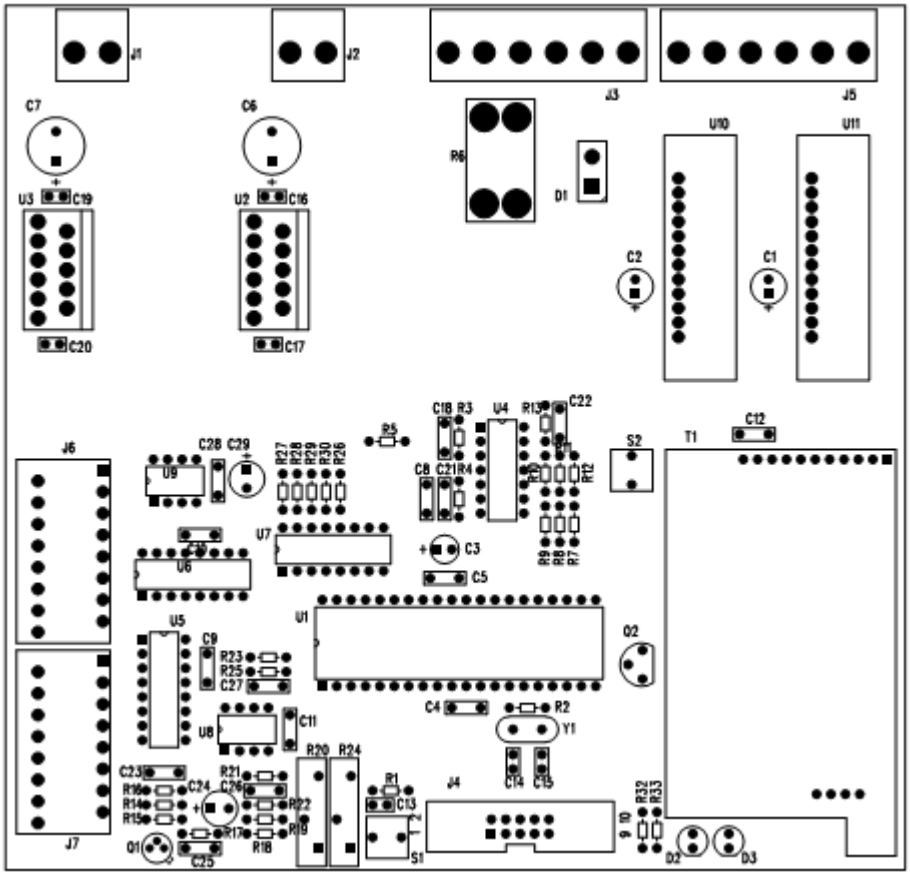

J1
J2
J3
J5
C7
C6
R6
D1
U10
U11
U3
C19
U2
C16
C2
C1
C20
C17
C18
R3
U4
R13
C22
R5
C12
J6
C28
C29
R27
R28
R29
R30
R26
S2
T1
U9
C8
C21
R4
R12
U7
C3
R9
R8
R7
U6
C5
U1
U5
C9
R23
R25
C27
Q2
C4
R2
Y1
U8
C11
R20
R24
C23
R21
R16
C24
C26
R14
R22
R15
R17
R19
R18
R1
C13
J4
C14
C15
R32
R33
J7
Q1
C25
S1
D2
D3

**Schéma 3**

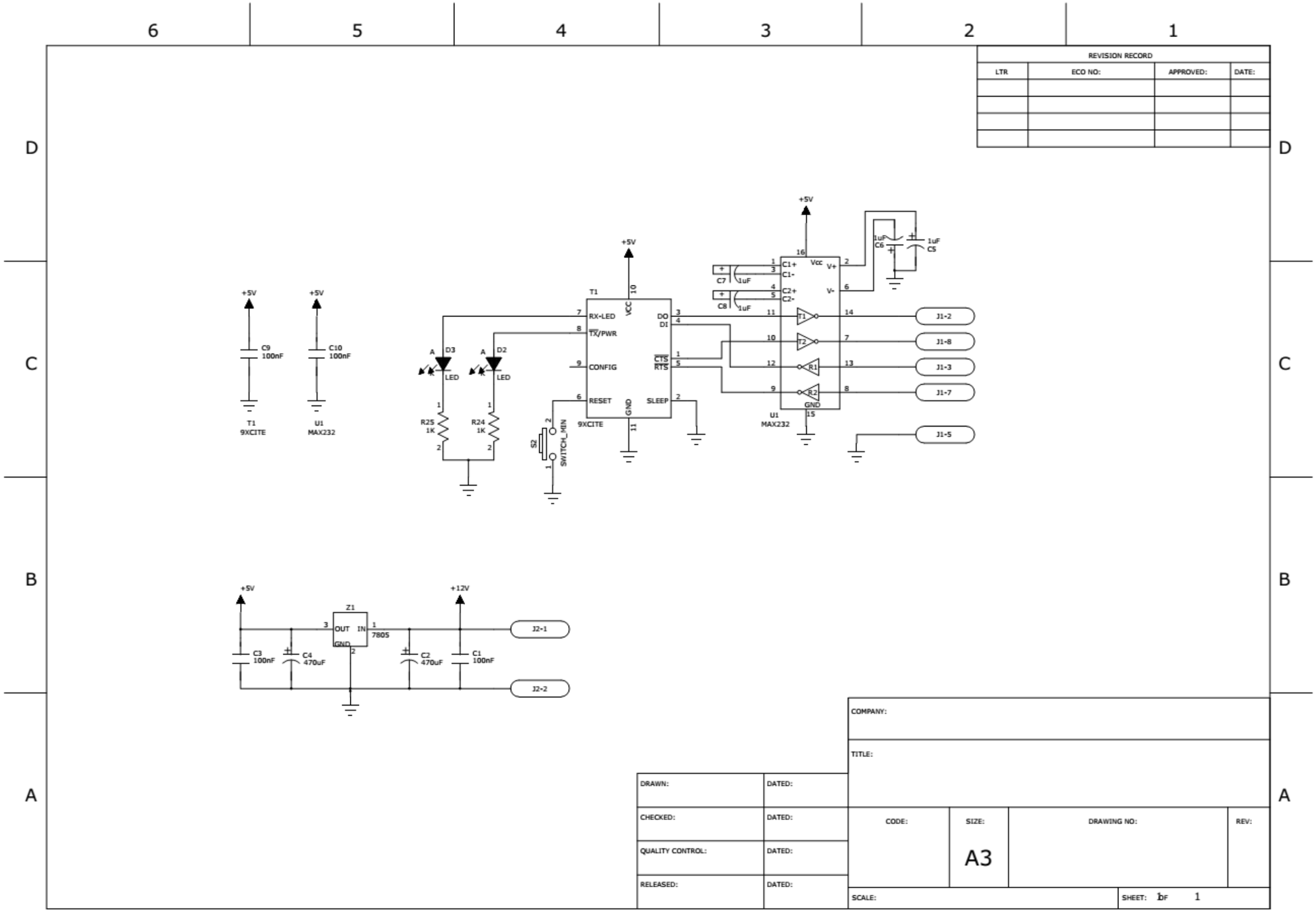
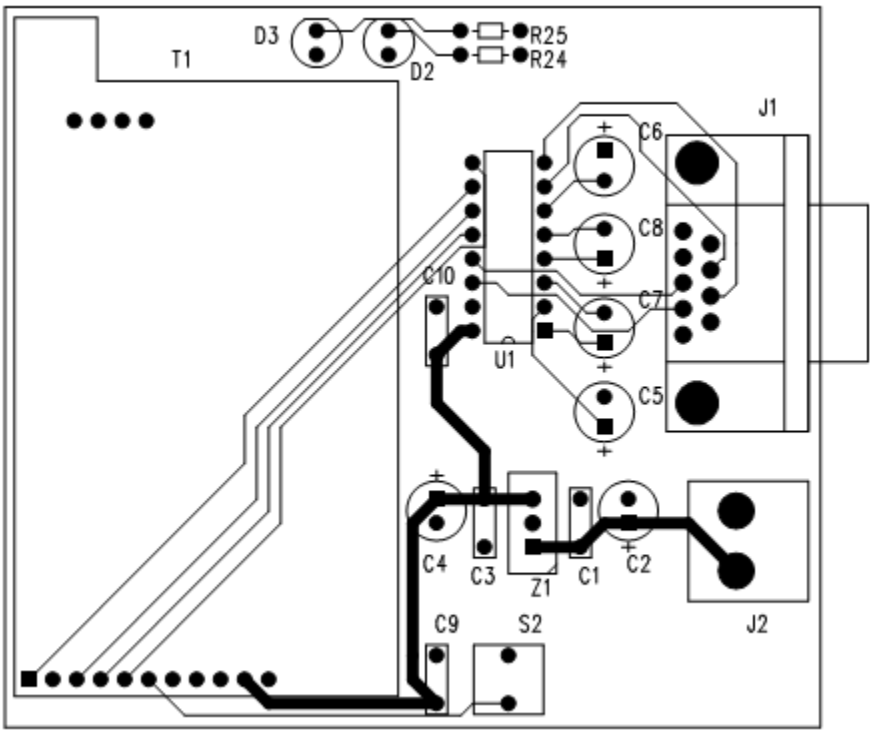

REVISION RECORD
LTR
ECO NO:
APPROVED:
DATE:
+5V
C9
100nF
C10
100nF
T1
9XCITE
U1
MAX232
D3
D2
LED
R25
1K
R24
1K
RX-LED
TX/PWR
CONFIG
RESET
VCC
GND
DO
DI
CTS
RTS
SLEEP
S2
SWITCH_MIN
C7
C8
1uF
C1+
C1-
C2+
C2-
Vcc
V+
V-
T1
T2
R1
R2
C6
C5
J1-2
J1-8
J1-3
J1-7
J1-5
+12V
Z1
OUT
IN
GND
7805
C3
100nF
C4
470uF
C2
470uF
C1
100nF
J2-1
J2-2
COMPANY:
TITLE:
DRAWN:
DATED:
CHECKED:
QUALITY CONTROL:
RELEASED:
CODE:
SIZE:
A3
DRAWING NO:
REV:
SCALE:
SHEET: 1 OF 1

**Schéma 4**



# Annexe 3 PROGRAMMES EN ASSEMBLEUR DU MICROCONTRÔLEUR

```
;*****************************************************************************************************************
; IRBOT - Robot mobile.
;
; RA
; 0 Entrée analogique, mesure niveau Courant moteur droite.
; 1 Entrée analogique, mesure niveau Courant moteur gauche.
; 2 Moteur feedback INDX.
; 3 Moteur feedback QEA.
; 4 Moteur feedback QEB.
; 5 Sélection Etat haut désactive la tension 12V.
;
; RB
; 0 PWM0 pour moteur droite.
; 1 PWM1 pour moteur droite/Freinage.
; 2 PWM2 pour moteur gauche.
; 3 PWM3 pour moteur gauche/Freinage.
; 4 Bit0 selection radar.
; 5 Bit1 selection radar.
; 6 Bit2 selection radar.
; 7 Clear to send.
;
; RC
; 0 Entrée détection obstacle.
; 1 .
; 2 PWM output (40KHz pour radar).
; 3 .
; 4 .
; 5 Request to send.
; 6 USART TX.
; 7 USART RX.
;
; RD
; 0 Sortie Direction moteur droite.
; 1 SPI SDO.
; 2 SPI SDI.
; 3 SPI SCK.
; 4 Motor driver temperature interruption.
; 5 Sélection  Etat bas démarre le ADG508A / réception radar.
; 6 Sélection Etat haut démarre le CD4051B / transmission radar.
; 7 Sortie Direction moteur gauche.
;
; RE
; 0 Entrée analogique mesure niveau Courant Batterie.
; 1 Entrée analogique mesure Tension Batterie.
; 2 Sortie pour mettre la transmission en état sleep.
;
;*****************************************************************************************************************
;*****************************************************************************************************************

        list p=18f4331,f=inhx8m


#include "p18f4331.inc"

;**** Constantes
*****************************************************************************************************************
;PortA
TRISA_Mask      equ B'00000011'         ; RA0 et RA1 entrées. les autres sorties.
PORTA_Mask      equ B'10000000'         ; Etat initial de la porte A.

;PortB
TRISB_Mask      equ B'10000000'         ; RB(0, 6) sorties, RB7 entrées.
PORTB_Mask      equ B'00000000'         ; Etat initial de la porte B.
Radar_Mask      equ B'00000111'         ; Mask pour les bits de sélection du radar.
```

```
;PortC
TRISC_Mask          equ B'10000001'         ; RC7/RX et RC0 entrée, les autres sorties.
PORTC_Mask          equ B'00100000'         ; Etat initial de la porte C.
PWM                 equ 2
TX                  equ 6

;PortD
TRISD_Mask          equ B'00010100'         ; RD2 and RD4 entrées, les autre sorties.
PORTD_Mask          equ B'00100000'         ; Etat initial de la porte D.
Rec_bit             equ 5                   ; 0 pour être capable de recevoire les signaux du radar.
Emet_bit            equ 6                   ; 1 pour être capable de transmetre par radar.
MD_Dir_Bit          equ 0                   ; 1 pour inverser la direction.
MG_Dir_Bit          equ 7                   ; 1 pour inverser la direction.

;PortE
TRISE_Mask          equ B'11111011'         ; RE2 entrée, les autre sorties.
PORTE_Mask          equ B'00000000'         ; Etat initial de la porte E.

;Etat
Etat_Normal         equ H'10'               ; Code d´état normal.
Int_Inc             equ H'11'               ; Code d'une interruption inconnue.

;Cmd
Cmd_Arret           equ H'10'               ; Commande d'arrêt.
Cmd_Démarrage       equ H'11'               ; Commande de démarrage.

;Valeur du timer
Dist_Max            equ H'FFFF' - 0xAA2D    ; Distance maximum du radar (43565 x 4 x 1/Fosc)= 8.713ms ou D = 2.86m (aller, retour)
avec Fosc = 20MHz et 43565 = 0xAA2D.
Pulse_delay         equ H'FFFF' - 0x09C4    ; Durée de .5ms pour émettre les signaux ultrason (2500 x 4 x 1/Fosc) = .5ms avec Fosc =
20MHz et 2500 = 0x09C4.
Silence_delay       equ H'FFFF' - 0x1388    ; Durée de 1ms pour mettre en état silence (5000 x 4 x 1/Fosc) = 1ms avec Fosc = 20MHz
et 5000 = 0x1388.
TMR1_compteur       equ 0xFFF6              ; FFFF - FFF6 = 9 -> Overflow apres 9 + 1 = 10 impulsions.

;Communication
TX_Num              equ D'21'               ; Nombre des octets a envoyer.

;*****************************************************************************************************************
;*****************************************************************************************************************

;****Variables*****************************************************************************************************
        cblock  0x00
                Etat, Distance:16, Mesure:4, TX_octet, RX_octet, Temp_RX, Radar_Drapot
        endc

; Etat:             Contient l'état du system.
; Distance:         Allocation de 16 octets pour les distances détectées par les 8 radars (2 octets chacune).
; Mesure:           Valeur lue par A/D: Courant moteur droite et gauche, Currant battérie, tension battérie.
; TX_octet:         Compteur des octets transmis par le port série.
; RX_octet:         Compteur des octets recus par le port série.
; Temp_RX:          Variable temporaire pour les octets recus par le port série.
; Radar_Drapot:     Drapot déterminant l'état du radar.

;*****************************************************************************************************************
;*****************************************************************************************************************
;*****Program******************************************************************************************************
        org     H'0'
        goto    init

        org     H'8'
        goto    ISR_Fonction

init
        ;movlw   0x7F
        ;movwf   OSCCON, 0       ; CPU configuré d'utiliser l'horloge interne durant le test du programme.
        clrf    BSR             ; Bank0.
        movlw   TRISA_Mask
```

```
        movwf   TRISA, 0            ; RA0 et RA1 entrées. les autres sorties.
        movlw   PORTB_Mask
        movwf   PORTB, 0
        movlw   TRISB_Mask
        movwf   TRISB, 0            ; RB(0, 6) sorties, RB7 entrées.
        movlw   PORTC_Mask
        movwf   PORTC, 0
        movlw   TRISC_Mask
        movwf   TRISC, 0            ; RC7/RX et RC0 entrée, les autres sorties.
        movlw   PORTD_Mask
        movwf   PORTD, 0
        movlw   TRISD_Mask
        movwf   TRISD, 0            ; RD2 and RD4 entrées, les autre sorties.
        movlw   PORTE_Mask
        movwf   PORTE, 0
        movlw   TRISE_Mask
        movwf   TRISE, 0            ;  RE2 entrée, les autre sorties.

init_ADCONV
        movlw   B'11000011'
        movwf   ANSEL0, 0           ; AN0, AN1, AN6 et AN7 sont des entrées analogiques, et AN(2-5) sont des I/O digitals.
        clrf    ANSEL1, 0           ; AN8 est une I/O digital.
        Movlw   B'01000100'
        movwf   ADCHS, 0            ; Channel AN0, AN1, AN6 et AN7 sont sélectés.
        movlw   H'3C'
        movwf   ADCON0, 0           ; Sélection Multi-channel groupe A, B, C et D d'une manière continue.
        clrf    ADCON1, 0           ; Tension de réference AVDD et AVSS.
        bsf     ADCON1, FIFOEN, 0   ; FIFO activé.
        movlw   H'7B'
        movwf   ADCON2, 0           ; Les résultats sont justifiées a gauche, oscillateur interne TAD = RC/4, temps d'échentillonage T =
64 TAD.
        movlw   H'90'
        movwf   ADCON3, 0           ; Interruption sur la 4ème convertion, la conversion est activée par software seulement.
        bcf     PIR1, ADIF, 0       ; Annuler le flag.
        bsf     PIE1, ADIE, 0       ; Activer l´interruption de AD.
        bsf     ADCON0, ADON, 0     ; Conversion A/D est en standby.

init_PWM_Moteur
        movlw   0x04
        movwf   PTCON0, 0           ; Horloge Fosc/4, Prescaler 1/4, Free running mode.
        clrf    PTCON1, 0           ; Désactiver PWM.
        clrf    PTPERH, 0
        movlw   D'99'
        movwf   PTPERL, 0           ; Fpwm = (Fosc/4)(1/4)/(99+1) = 12.500KHz avec Fosc = 20MHz.
        clrf    PDC0H, 0
        clrf    PDC0L, 0            ; Rapport cyclique 0% (Vitesse = 0), moteur droite.
        clrf    PDC1H, 0
        clrf    PDC1L, 0            ; Rapport cyclique 0% (Vitesse = 0), moteur gauche.
        movlw   H'30'
        movwf   PWMCON0, 0          ; PWM0, PWM1, PWM2 et PWM3 sont des sorties PWM avec (PWM0 PWM1) et (PWM2 PWM3) en
mode complément.
        clrf    PWMCON1, 0          ; Default configuration.
        clrf    OVDCONS, 0          ; En cas de baypass, les sorties sont désactivées.

init_USART
        clrf    TX_octet, 0         ; Annuler le compteur des octets transmis.
        clrf    RX_octet, 0         ; Annuler le compteur des octets reçus.
        movlw   Etat_Normal         ; Etat normal.
        movwf   Etat, 0
        bsf     BAUDCTL, BRG16, 0   ; Utiliser SPBRGH et SPBRG pour la vitesse en baud.
        movlw   0x10
        movwf   SPBRGH, 0
        movlw   0x45
        movwf   SPBRG, 0            ; Baud rate est de 1200 baud avec Fosc = 20MHz.
        movlw   0x24
        movwf   TXSTA, 0            ; TX 8-bit, TX activer, mode asynchrone, haute vitesse.
        movlw   0x90
        movwf   RCSTA, 0            ; RX 8-bit, RX activer, porte série activée.
```

```
        bcf     PIR1, RCIF, 0       ; Annuler le flag.
        bcf     PIR1, TXIF, 0       ; Annuler le flag.
        bsf     PIE1, RCIE, 0       ; Activer l´interruption de RX.
        bcf     PIE1, TXIE, 0       ; Désactiver l´interruption de TX.

init_PWM_Radar                      ; PWM sur CCP1 utilisant TMR2.
init_TMR2
        clrf    T2CON, 0            ; Prescale est 1:1, incrCmenter TMR2 chaque 4*1/Fosc unité de temps, TMR2 en standby.
        clrf    TMR2, 0             ; Annuler TMR2.
        movlw   D'62'
        movwf   CCPR1L, 0           ; Duty cycle 50%.
init_CCP1
        movlw   D'124'
        movwf   PR2, 0              ; Période et de (4*1/Fosc)*(124+1).
        clrf    CCP1CON, 0          ; PWM désactiver, 2 LSbs de Duty cycle = 00.

init_Det_Radar                      ; Entrée détection du signal radar utilisant TMR1.
init_TMR1
        movlw   H'86'
        movwf   T1CON, 0            ; Configuration sur 16 bits, horloge sur RC0 entrée synchronisée, TMR1 désactivé.
        bsf     T1CON, T1SYNC, 0
        movlw   high TMR1_compteur
        movwf   TMR1H, 0
        movlw   low TMR1_compteur
        movwf   TMR1L, 0            ; Initialiser TMR1 par le nombre de détection.
        bcf     PIR1, TMR1IF, 0     ; Annuler le drapot.
        bcf     PIE1, TMR1IE, 0     ; Interruption sur TMR1 overflow est désactivée.

init_TMR0
        movlw   H'08'
        movwf   T0CON, 0            ; Arrêter TMR0, configuration sur 16 bits, horloge interieur, sans diviseur d'horloge.
        movlw   high Dist_Max
        movwf   TMR0H, 0
        movlw   low Dist_Max
        movwf   TMR0L, 0            ; Initialiser TMR0 par la temporisation max.
        bcf     INTCON, TMR0IF, 0   ; Annuler le flag.
        bsf     INTCON, TMR0IE, 0   ; Interruption sur TMR0 overflow.

init_Interruption
        clrwdt
        bsf     RCON, POR, 0
        bsf     RCON, BOR, 0
        bcf     RCON, IPEN, 0       ; Interruption par non priorité.
        bsf     INTCON, GIEH, 0     ; Activer l´interruption.
        bsf     INTCON, GIEL, 0     ; Activer l´interruption des périférikues.


        goto    Main

;*****************************************************************************************************************
;*****************************************************************************************************************


;**************ISR_Fonction*****************************************************************************************
ISR_Fonction
        btfsc   INTCON, TMR0IF, 0   ; TMR0 overflow?
        goto    TMR0_INT            ; Oui.
        btfsc   PIR1, TMR1IF, 0     ; Non, TMR1 overflow?
        goto    TMR1_INT            ; Oui.
        btfsc   PIR1, ADIF, 0       ; Non, AD interruption?
        goto    AD_INT              ; Oui.
        btfsc   PIR1, RCIF, 0       ; Non, USART RX interruption?
        goto    USART_RX_INT        ; Oui.
        btfsc   PIR1, TXIF, 0       ; Non, USART TX interruption?
        goto    USART_TX_INT        ; Oui.
        goto    INT_INCONNUE        ; Non, interruption inconnue.
```

```
;****************TMR0_ISR*****************************************
TMR0_INT
        btfss   T2CON, TMR2ON, 0    ; PWM radar est activé?
        bra     Etat_PWM            ; Non, teste Etat.
        bcf     T2CON, TMR2ON, 0    ; Oui, Désactiver le.
        clrf    CCP1CON, 0
        bcf     PORTC, PWM, 0       ; Annuler la sortie PWM.
        incf    Radar_Drapot,1 ,0   ; Etat Silence
        movlw   high Silence_delay
        movwf   TMR0H, 0
        movlw   low Silence_delay
        movwf   TMR0L, 0            ; Initialiser TMR0 par la temporisation d'état silence.
        bra     FIN_TMR0_ISR        ; Fin TMR0 ISR.

Etat_PWM
        btfsc   Radar_Drapot, 0, 0  ; Etat Silence?
        bra     Ecoute_PWM          ; Oui mettre en écoute.
        bsf     PORTD, Rec_bit, 0   ; Non Désactiver la réception radar.
        bcf     T1CON, TMR1ON, 0    ; Désactiver TMR1.
        bcf     PIR1, TMR1IF, 0     ; Annuler le drapot.
        bcf     PIE1, TMR1IE, 0     ; Interruption sur TMR1 overflow est désactivée.
        movlw   H'10'
        addwf   PORTB, 1, 0         ; Choisiser l'émeteur suivant.
        movlw   high Pulse_delay
        movwf   TMR0H, 0
        movlw   low Pulse_delay
        movwf   TMR0L, 0            ; Initialiser TMR0 par la temporisation d'impulsion.
        movlw   H'0C'               ; Activer PWM du radar.
        movwf   CCP1CON, 0
        bsf     T2CON, TMR2ON
        bra     FIN_TMR0_ISR        ; Fin TMR0 ISR.

Ecoute_PWM
        clrf    Radar_Drapot, 0     ; Etat d'écoute.
        bcf     PORTD, Rec_bit, 0   ; Activer la réception radar.
        movlw   high Dist_Max
        movwf   TMR0H, 0
        movlw   low Dist_Max
        movwf   TMR0L, 0            ; Initialiser TMR0 par la temporisation max.
        movlw   high TMR1_compteur
        movwf   TMR1H, 0
        movlw   low TMR1_compteur
        movwf   TMR1L, 0            ; Initialiser TMR1 par le nombre de détection.
        bsf     T1CON, TMR1ON       ; Activer TMR1.
        bcf     PIR1, TMR1IF, 0     ; Annuler le drapot.
        bsf     PIE1, TMR1IE, 0     ; Interruption sur TMR1 overflow est activée.

FIN_TMR0_ISR
        bcf     INTCON, TMR0IF, 0   ; Annuler le drapot de TMR0.
        goto    ISR_FIN             ; Fin.


;****************TMR1_ISR*****************************************
TMR1_INT
        swapf   PORTB, 0, 0         ; Lire l'émeteur.
        andlw   Radar_Mask
        lfsr    FSR0, Distance      ; Initialiser le pointeur/enregistrer le temp.
        rlncf   WREG, 0, 0          ; Calculer l'offset nécessaire.
        movff   TMR0L, PLUSW0       ; Enregistrement.
        addlw   1
        movff   TMR0H, PLUSW0
        bsf     PORTC, 4, 0

FIN_TMR1_INT
        bcf     T1CON, TMR1ON, 0    ; Désactiver TMR1.
        bcf     PIR1, TMR1IF, 0     ; Annuler le drapot de TMR1.
        bcf     PIE1, TMR1IE, 0     ; Interruption sur TMR1 overflow est désactivée.
        goto    ISR_FIN             ; Fin.
```

```
;**************USART_RX_ISR****************************************
USART_RX_INT
        movff   RCREG, Temp_RX      ; Enregistré l'octet reçu.
        movf    RX_octet, 0, 0      ; Information de controle?
        bnz     RX_DATA             ; Oui, configurer le system.
        movlw   Cmd_Arret
        subwf   Temp_RX, 0, 0       ; Non, commande d'arret?
        bz      RX_Arret            ; Oui, arreté le systeme.
        movlw   Cmd_Démarrage
        subwf   Temp_RX, 0, 0       ; Non, commande de démarrage?
        bz      RX_Activer          ; Oui, démarrer le systeme.
        bra     FIN_USART_RX_ISR    ; Non, fin de l'interruption.

RX_Arret
        btfss   T0CON, TMR0ON, 0    ; System activé?
        bra     FIN_USART_RX_ISR    ; Non, fin de l'interruption.
        bcf     T0CON, TMR0ON, 0    ; Oui, désactivé TMR0.
        bcf     T1CON, TMR1ON, 0    ; Désactivé TMR1.
        bcf     ADCON0, GO_DONE, 0          ; Désactiver la convertion A/N.
        bcf     PTCON1, PTEN, 0     ; Désactiver PWM des moteurs.
        bsf     PORTD, Rec_bit, 0   ; Désactiver la réception radar.
        bcf     PORTD, Emet_bit, 0  ; Désactiver la transmition radar.
        bcf     T2CON, TMR2ON       ; Désactiver PWM du radar.
        bra     FIN_USART_RX_ISR    ; Fin de l'interruption.

RX_Activer
        btfsc   T0CON, TMR0ON, 0    ; System désactivé?
        bra     RX_attendre         ; Non, attendre les inforation de contrôle.
        bsf     ADCON0, GO_DONE, 0          ; Oui, activer la convertion A/N.
        bsf     PORTD, Emet_bit, 0  ; Activer la transmition radar.
        bsf     T0CON, TMR0ON, 0    ; Activer TMR0.
        bsf     PTCON1, PTEN, 0     ; Activer PWM des moteurs.

RX_attendre
        movlw   B'00001100'         ; Attendre les informations de contrôle.
        movwf   RX_octet, 0
        bra     FIN_USART_RX_ISR    ; Fin de l'interruption.

RX_DATA
        btfsc   RX_octet, 2, 0      ; C'est le deuxième octet reçu?
        bra     Moteur_Droite       ; Oui, configurer la vitesse du moteur droite.
        btfsc   RX_octet, 3, 0      ; Non, c'est le troisième octet reçu?
        bra     Moteur_Gauche       ; Oui, configurer la vitesse du moteur gauche.
        clrf    RX_octet, 0         ; Non, annuler le compteur.
        bra     Activation_TX       ; Activer la transmission.

Moteur_Droite
        movf    Temp_RX, 0, 0       ; Tester la valeur.
        Bz      VD_Zero             ; Nombre égale à zéro.
        bnn     VD_Positive         ; Nombre positive.
        negf    Temp_RX             ; Nombre négatif, négation vitesse.
        bsf     PORTD, MD_Dir_Bit, 0        ; Moteur en arrier.
        bra     VD_Conf

VD_Positive
        bcf     PORTD, MD_Dir_Bit, 0        ; Moteur en avant.

VD_Conf
        movlw   H'04'
        mulwf   Temp_RX, 0          ; Déviation vers la gauche par 2.
        movff   PRODH, PDC0H
        movff   PRODL, PDC0L        ; Configurer la vitesse.
        bsf     OVDCOND, POVD0, 0           ; Désactiver la frénage.
        bra             VD_Fin

VD_Zero
        bcf     OVDCOND, POVD0, 0           ; Activer la frénage.
```

```
VD_Fin
        bcf     RX_octet, 2, 0          ; Le deuxième octet est reçu.
        bra     FIN_USART_RX_ISR        ; Fin de l'interruption.

Moteur_Gauche
        movf    Temp_RX, 0, 0           ; Tester la valeur.
        Bz      VG_Zero                 ; Nombre égale à zéro.
        bnn     VG_Positive             ; Nombre positive.
        negf    Temp_RX                 ; Nombre négatif, négation vitesse.
        bsf     PORTD, MG_Dir_Bit, 0            ; Moteur en arrier.
        bra     VG_Conf

VG_Positive
        bcf     PORTD, MG_Dir_Bit, 0            ; Moteur en avant.

VG_Conf
        movlw   H'04'
        mulwf   Temp_RX, 0              ; Déviation vers la gauche par 2.
        movff   PRODH, PDC1H
        movff   PRODL, PDC1L            ; Configurer la vitesse.
        bsf     OVDCOND, POVD2, 0               ; Désactiver la frénage.
        bra     VG_Fin

VG_Zero
        bcf     OVDCOND, POVD2, 0               ; Activer la frénage.

VG_Fin
        bcf     RX_octet, 3, 0          ; Le troisième octet est reçu.

Activation_TX
        movlw   TX_Num                  ; Initialise le compteur des octets a transmetre.
        movwf   TX_octet, 0
        lfsr    FSR1, Etat              ; Initialise le pointeur.
        movff   POSTINC1, TXREG         ; Envoyer le premier octet.
        decf    TX_octet, 1, 0          ; Décrémenter le compteur.
        bcf     PIR1, TXIF, 0           ; Annuler le drapot de TX.
        bsf     PIE1, TXIE, 0           ; Activer l´interruption de TX.

FIN_USART_RX_ISR
        bcf     PIR1, RCIF, 0           ; Annuler le drapot de RX.
        goto    ISR_FIN                 ; Fin.

;**************USART_TX_ISR****************************************
USART_TX_INT
        tstfsz  TX_octet,0              ; Informations a envoyer?
        bra     Envoie_Inf              ; Oui, continuer a envoyer.

Desactiver_TX
        bcf     PIE1, TXIE, 0           ; Non, désactiver l´interruption de TX.
        bra     FIN_USART_TX_ISR        ; Fin de l'interruption.

Envoie_Inf
        movff   POSTINC1, TXREG         ; Envoyer l'octet.
        decf    TX_octet, 1, 0          ; Decrementer le compteur.

FIN_USART_TX_ISR
        bcf     PIR1, TXIF, 0           ; Annuler le flag.
        goto    ISR_FIN                 ; Fin.

;****************AD_ISR*****************************************
AD_INT
        lfsr    FSR2, Mesure            ; Initialise le pointeur.
        movff   ADRESH, POSTINC2        ; Enregistrement le courant MD.
        movff   ADRESH, POSTINC2        ; Enregistrement le courant MG.
        movff   ADRESH, POSTINC2        ; Enregistrement le courant de la Battérie.
        movff   ADRESH, POSTINC2        ; Enregistrement la tension de la Battérie.

FIN_AD_INT
```

```
        bcf     PIR1, ADIF, 0       ; Annuler le drapot
        goto    ISR_FIN

;****************INT_INCONNUE_ISR*********************************
INT_INCONNUE
        bcf     INTCON, RBIF, 0     ; Annuler le drapot
        bcf     INTCON, TMR0IF, 0   ; Annuler le drapot de TMR0.
        bcf     INTCON3, INT1IF, 0  ; Annuler le drapot de INT1.
        movlw   Int_Inc             ; Code d'erreur.
        movwf   Etat, 0

ISR_FIN
        retfie

;*************************************************************************************************************
;*************************************************************************************************************

;************Main*********************************************************************************************
Main

        bra     Main
        end
```

# RÉFÉRENCES


## 1. POUR LES RÉSEAUX DE NEURONES.

1.1. Principles of Neurocomputing for Science and Engineering / Fredric M. Ham, Ivica Kostanic / McGraw-Hill / ISBN : 0-07-025966-6.

1.2. Bio-Inspired Computing Machines / Daniel Mange, Marco Tomassini / ISBN : 2-88074-371-0.

1.3. Mathlab Help: Neural Network Toolbox.

## 2. POUR L'ELECTRONIQUE.

2.1. Embedded Microcontrollers / Todd D. Morton / ISBN : 0-13-907577-1.

2.2. MICROCHIP. http://www.microchip.com

2.3. MAXIM. http://www.maxim-ic.com

2.4. NATIONAL SEMICONDUCTOR. www.national.com

2.5. TEXAS INSTRUMENTS. http://www.ti.com